

PAT-NO: JP411308255A

DOCUMENT-IDENTIFIER: JP 11308255 A

TITLE: SYSTEM, METHOD AND DEVICE FOR DATA COMMUNICATION AND STORAGE MEDIUM

PUBN-DATE: November 5, 1999

INVENTOR-INFORMATION:

NAME	COUNTRY
ONISHI, SHINJI	N/A
HATAE, SHINICHI	N/A
ARAI,DA, MITSUHIISA	N/A
KOBAYASHI, TAKASHI	N/A

ASSIGNEE-INFORMATION:

NAME	COUNTRY
CANON INC	N/A

APPL-NO: JP10111681

APPL-DATE: April 22, 1998

INT-CL (IPC): H04L012/40, G06F013/00 , H04L029/08

ABSTRACT:

PROBLEM TO BE SOLVED: To make it possible to easily and quickly transfer data and to surely execute data transfer by performing transmission/reception of information data in accordance with the lowest reception capability out of reception capabilities of one or more reception devices.

SOLUTION: A computer 10, a VCR 28 and a printer 60 constitute a node of a 1394 serial bus through first to third 1394 interfaces 14, 44 and 62 and, at the same time, are mutually connected by way of the 1394 interfaces 14, 44 and 62, and are constituted so that transmission/reception of the data, control by a command and the like are enabled. In a data communication system like this, it is designed to perform transmission/reception of information data in accordance with the lowest reception capability out of reception capabilities of one or more reception devices. Thus, irrespective of a buffer size provided to each destination, source can execute the data transfer in the size unit below a notified buffer size and so can surely execute communication.

COPYRIGHT: (C)1999,JPO

(51)IntCl. ⁹	識別記号	F I	
H 0 4 L 12/40		H 0 4 L 11/00	3 2 0
G 0 6 F 13/00	3 5 1	G 0 6 F 13/00	3 5 1 A
H 0 4 L 29/08		H 0 4 L 13/00	3 0 7 Z

審査請求 未請求 請求項の数14 O L (全 32 頁)

(21)出願番号 特願平10-111681

(22)出願日 平成10年(1998)4月22日

(71)出願人 000001007

キヤノン株式会社

東京都大田区下丸子3丁目30番2号

(72)発明者 大西 慎二

東京都大田区下丸子3丁目30番2号 キヤノン株式会社内

(72)発明者 波多江 真一

東京都大田区下丸子3丁目30番2号 キヤノン株式会社内

(72)発明者 新井田 光央

東京都大田区下丸子3丁目30番2号 キヤノン株式会社内

(74)代理人 弁理士 園分 孝悦

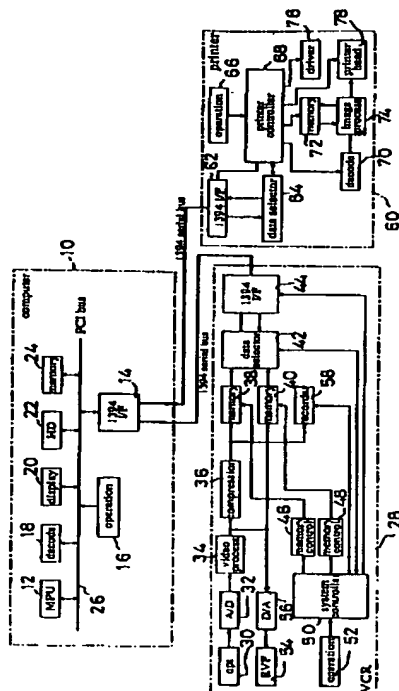
最終頁に続く

(54)【発明の名称】 データ通信システム、方法及び装置並びに記憶媒体

(57)【要約】

【課題】 簡便にかつ高速にデータを転送できるようにするとともに、確実にデータ転送を実行できるようにする。

【解決手段】 所定の機器に固有の固有情報と、送信機器と1つ以上の受信機器との間の論理的な接続を示すコネクション情報とを用いて、情報データの送受信を行うデータ通信システムにおいて、上記1つ以上の受信機器の受信能力のうち、最も低い受信能力に従って上記情報データの送受信を行うようにすることにより、各 destinations に設けられているバッファサイズに関係なく、ソースは通知されたバッファサイズ以下のサイズ単位でデータ転送を行い、通信を確実に実行することが可能となる。



【特許請求の範囲】

【請求項1】 所定の機器に固有の固有情報と、送信機器と1つ以上の受信機器との間の論理的な接続を示すコネクション情報とを用いて、情報データの送受信を行うデータ通信システムにおいて、

上記1つ以上の受信機器の受信能力のうち、最も低い受信能力に従って上記情報データの送受信を行うことを特徴とするデータ通信システム。

【請求項2】 上記所定の機器は、上記コネクション情報を設定する管理機器であることを特徴とする請求項1に記載のデータ通信システム。

【請求項3】 上記管理機器は、上記情報データの送信に先立って、上記固有情報と上記コネクション情報とを上記送信機器と上記1つ以上の受信機器とに通知することを特徴とする請求項1または2に記載のデータ通信システム。

【請求項4】 上記送信機器と上記1つ以上の受信機器とは、上記管理機器を介することなく、上記固有情報と上記コネクション情報とを用いて上記情報データの送受信を行うことを特徴とする請求項2または3に記載のデータ通信システム。

【請求項5】 上記データ通信システムの接続構成が初期化された後、上記送信機器と上記1つ以上の受信機器とは、上記管理機器を介することなく上記情報データの送受信を再開することを特徴とする請求項1～4の何れか1項に記載のデータ通信システム。

【請求項6】 上記データ通信システムの接続構成が初期化された後、上記送信機器と上記1つ以上の受信機器とは、上記固有情報と上記コネクション情報とを用いて上記情報データの送受信を再開することを特徴とする請求項1～5の何れか1項に記載のデータ通信システム。

【請求項7】 上記受信機器の受信能力は、上記受信機器の具備する受信バッファのサイズを含むことを特徴とする請求項1～6の何れか1項に記載のデータ通信システム。

【請求項8】 上記管理機器は、上記送信機器から送信される情報データを受信する受信機器の数を、上記コネクション情報と共に管理することを特徴とする請求項2または3に記載のデータ通信システム。

【請求項9】 上記情報データは、特定の受信機器を指定しない情報と共に送信されることを特徴とする請求項1～8の何れか1項に記載のデータ通信システム。

【請求項10】 上記固有情報と上記コネクション情報とは、上記データ通信システムの接続構成が初期化されても変化しないことを特徴とする請求項1～9の何れか1項に記載のデータ通信システム。

【請求項11】 所定の機器に固有の固有情報と、送信機器と1つ以上の受信機器との間の論理的な接続を示すコネクション情報とを用いて、情報データの送受信を行うデータ通信システムに適用可能なデータ通信方法にお

いて、

上記1つ以上の受信機器の受信能力のうち、最も低い受信能力に従って上記情報データの送受信を実行させることを特徴とするデータ通信方法。

【請求項12】 所定の機器に固有の固有情報と、1つ以上の受信機器との間の論理的な接続を示すコネクション情報とを用いて情報データの送受信を行う通信手段と、

上記1つ以上の受信機器の受信能力のうち、最も低い受信能力に従って上記通信手段の動作を制御する制御手段とを具備することを特徴とするデータ通信装置。

【請求項13】 請求項11に記載のデータ通信方法の手順をコンピュータに実行させるためのプログラムを格納したことを特徴とする記憶媒体。

【請求項14】 請求項12に記載の各手段としてコンピュータを機能させるためのプログラムを格納したことを特徴とする記憶媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明はデータ通信システム、方法及び装置並びに記憶媒体に関し、特に、制御信号とデータを混在させて通信することが可能なデータ通信バスを用いて複数電子機器（以下、機器）間を接続して、各機器間でデータ通信を行うシステムを所有する装置に用いて好適なものである。

【0002】

【従来の技術】パソコン周辺機器の中で、最も利用頻度が高いのはハードディスクやプリンタであり、これらの周辺装置は小型コンピュータ用汎用型インターフェイスで代表的なデジタルインターフェイス（以下、デジタルI/F）であるSCSI等を介してパソコン間との接続がなされ、データ通信が行われている。

【0003】また、デジタルカメラやデジタルビデオカメラといった記録再生装置もパソコン（以下、PC）への入力手段として用いられる周辺装置の1つであり、近年、デジタルカメラやビデオカメラで撮影した静止画や動画といった映像をパソコンPCへ取り込み、ハードディスクに記憶したり、またはパソコンPCで編集した後、プリンタでカラープリントするといった分野の技術が進んでおり、ユーザーも増えている。

【0004】取り込んだ画像データをパソコンPCからプリンタやハードディスクへ出力する際などに、上記のSCSI等を經由してデータ通信がされるものであり、そのようなときに画像データのようなデータ量の多い情報を送るためにも、こういったデジタルI/Fには転送データレートの高い、かつ汎用性のあるものが必要とされる。

【0005】図21に、従来の例としてデジタルカメラ、パソコンPC及びプリンタを接続したときのブロック図を示す。図21において、101はデジタルカメラ、102はパソコン(PC)、103はプリンタである。さらに、10

4 はデジタルカメラの記録部であるメモリ、105 は画像データの復号化回路、106 は画像処理部、107 はD/A コンバータ、108 は表示部であるEVF、109 はデジタルカメラのデジタルI/O 部、110 はパソコンPCのデジタルカメラとのデジタルI/O 部、111 はキーボードやマウスなどの操作部である。

【0006】112 は画像データの復号化回路、113 はディスプレイ、114 はハードディスク装置、115 はRAM 等のメモリ、116 は演算処理部のMPU、117 はPCI バス、118 はデジタルI/F のSCSI インタフェース(ボード)、119 はパソコンPCとSCSI ケーブルで繋がったプリンタのSCSI インタフェース、120 はメモリ、121 はプリンタヘッド、122 はプリンタ制御部のプリンタコントローラ、123 はドライバである。

【0007】まず、デジタルカメラで撮像した画像をパソコンPCに取り込み、またパソコンPCからプリンタへ出力するときの手順の説明を行う。デジタルカメラ101 のメモリ104 に記憶されている画像データが読み出されると、上記読み出された画像データのうちの一方は復号化回路105 で復号化され、画像処理回路106 で表示するための画像処理がなされ、D/A コンバータ107 を経て、EVF108 で表示される。

【0008】また一方では、外部出力するためにデジタルI/O 部109 から、ケーブルを伝ってPC102 のデジタルI/O 部110 へ至る。PC102 内では、PCI バス117 を相互伝送のバスとして、デジタルI/O 部110 から入力した画像データは、記憶する場合はハードディスク114 で記憶され、表示する場合は復号化回路112 で復号化された後、メモリ115 で表示画像としてメモリされて、ディスプレイ113 でアナログ信号に変換されてから表示される。パソコンPC102 での編集時等の操作入力は操作部111 から行い、パソコンPC102 全体の処理はMPU116で行う。

【0009】また、画像をプリント出力する際は、パソコンPC102 内のSCSI インタフェースボード118 から画像データをSCSI ケーブルを介して伝送し、プリンタ103 側のSCSI インタフェース119 で受信し、メモリ120 でプリント画像として形成され、プリンタコントローラ122 の制御でプリンタヘッド121 とドライバ123 が動作して、メモリ120 から読み出したプリント画像データをプリントする。

【0010】以上が、従来の画像データをパソコンPC取り込み、またはプリントするまでの手順である。このように、従来はホストであるパソコンPCにそれぞれの機器が接続され、パソコンPCを介してから、記録再生装置で撮像した画像データをプリントしている。

【0011】また、デジタルVTR、TV、チューナなどのAV機器や、パーソナルコンピュータ(以下、PCと称する)等をIEEE1394 シリアルバス(以下、1394と称する)を用いて相互に接続し、これらの間においてディ

タルビデオ信号、ディジタルオーディオ信号などを送受信するデータ通信システムが提案されている。

【0012】これらのシステムにおいては、リアルタイムにデータ転送することが重要となるため、いわゆる同期通信(以下、Isochronous 通信と称する)によって、データ通信を行なっている。この場合には、データ転送のリアルタイム性は保証されるが、通信が確実に実行されることは保証されない。

【0013】

【発明が解決しようとする課題】しかしながら、上記従来例で挙げたデジタルインターフェイスの問題点として、SCSIには転送データレートの低いものや、パラレル通信のためケーブルが太いもの、接続される周辺機器の種類や数、接続方式などにも制限があり、多くの面での不利性も指摘されている。

【0014】また、従来の1394通信の場合には、同期通信を行なうため、通信が確実に実行されることは保証されない。したがって、確実にデータ転送を行ないたい場合には、従来の1394 Isochronous通信を使用することはできなかった。

【0015】また、従来の1394 Isochronous通信では、通信帯域に空きがある場合にも、通信の総数が64に制限される。このため、通信帯域をあまり要求しないような通信を多数行ないたい場合には、従来の1394 Isochronous通信を使用することはできないといった問題点があった。

【0016】また、従来の1394通信方式では、データ転送の間に、バスリセットやエラーによる、データ転送の中断が生じることが考えられる。この場合、従来の1394通信方式では、どのようなデータ内容が失われたのかを知ることができない問題があった。そのため、従来の1394通信方式では、該データ転送中断からの復帰を行なうためには、非常に複雑な通信手順を踏むことを要求されるという問題点があった。

【0017】本発明は、上記問題点を解決するためになされたもので、従来の通信方式の不利便性を解決し、簡便にかつ高速にデータを転送できるようにするとともに、確実にデータ転送を実行できるようにすることを第1の目的とする。また、本発明は、通信帯域をあまり使用しない場合に、多数の通信を同時に行なえるようにすることを第2の目的とする。また、本発明は、データ転送中断により失われたデータを容易に検出することが可能で、上記データ転送中断からの復帰を、確実に、かつ簡単に行なえるようにすることを第3の目的とする。また、本発明は、情報データの転送を管理する複数のコントロールノードがネットワーク上に存在する場合に、個々のコントロールノードにより管理される通信を識別できる機能を提供することを第4の目的とする。さらに、本発明は、同じ情報データを1つのソースノードから複数のデスティネーションノードに対して転送することの

できる機能を提供することを第4の目的とする。

【0018】

【課題を解決するための手段】従来抱えている問題を解決するために、本発明は、従来からあるデジタルI/Fの問題点を極力解消した、各デジタル機器に統一されて搭載されるような汎用型デジタルI/F(例えばIEEE1394-1995 ハイパフォーマンス・シリアルバス)を用いて、パソコンPCやプリンタ、その他周辺装置、またデジタルカメラやデジタルVTRの記録再生装置等をネットワーク構成で接続したときの機器間データ通信を実現し、記録再生装置からビデオデータ等のパソコンPCへの取り込み、また、映像データをプリンタへ直接転送しプリントなどを実現する。このようなネットワークにおいて、各種のデータをAsynchronousトランザクションによりそれぞれのデータを複数に分割して伝送するプロトコルを提供するものである。ペイロード内に、バスリセット等によっても変化しない、ソースノード固有のノード情報とソースノードが管理するID番号を組み合わせた識別データを付加する。コントローラノードは、ソースに対して論理的に接続されたデスティネーション数を通知する。コントローラノードはデスティネーションノードが複数存在する場合は、各デスティネーションノードが持つバッファサイズの中で最小のバッファサイズをソースノードに通知する。ソースノードは、コントローラノードから通知されたデスティネーションノードのバッファサイズごとに該デスティネーションからの受信確認応答パケットをまって、次のデータパケットを送信する。ソースノードからの送信データ終了を示すパケットに対してそれぞれのデスティネーションは、受信確認応答パケットを返すようにしている。

【0019】本発明のデータ通信システムは、所定の機器に固有の固有情報と、送信機器と1つ以上の受信機器との間の論理的な接続を示すコネクション情報とを用いて、情報データの送受信を行うデータ通信システムにおいて、上記1つ以上の受信機器の受信能力のうち、最も低い受信能力に従って上記情報データの送受信を行うことを特徴としている。また、本発明のデータ通信システムの他の特徴とするところは、上記所定の機器は、上記コネクション情報を設定する管理機器であることを特徴としている。また、本発明のデータ通信システムのその他の特徴とするところは、上記管理機器は、上記情報データの送信に先立って、上記固有情報と上記コネクション情報とを上記送信機器と上記1つ以上の受信機器とに通知することを特徴としている。また、本発明のデータ通信システムのその他の特徴とするところは、上記送信機器と上記1つ以上の受信機器とは、上記管理機器を介することなく、上記固有情報と上記コネクション情報とを用いて上記情報データの送受信を行うことを特徴としている。また、本発明のデータ通信システムのその他の特徴とするところは、上記データ通信システムの接続構

成が初期化された後、上記送信機器と上記1つ以上の受信機器とは、上記管理機器を介することなく上記情報データの送受信を再開することを特徴としている。また、本発明のデータ通信システムのその他の特徴とするところは、上記データ通信システムの接続構成が初期化された後、上記送信機器と上記1つ以上の受信機器とは、上記固有情報と上記コネクション情報とを用いて上記情報データの送受信を再開することを特徴としている。また、本発明のデータ通信システムのその他の特徴とするところは、上記受信機器の受信能力は、上記受信機器の具備する受信バッファのサイズを含むことを特徴としている。また、本発明のデータ通信システムのその他の特徴とするところは、上記管理機器は、上記送信機器から送信される情報データを受信する受信機器の数を、上記コネクション情報と共に管理することを特徴としている。また、本発明のデータ通信システムのその他の特徴とするところは、上記情報データは、特定の受信機器を指定しない情報と共に送信されることを特徴としている。また、本発明のデータ通信システムのその他の特徴とするところは、上記固有情報と上記コネクション情報とは、上記データ通信システムの接続構成が初期化されても変化しないことを特徴としている。

【0020】本発明のデータ通信方法は、所定の機器に固有の固有情報と、送信機器と1つ以上の受信機器との間の論理的な接続を示すコネクション情報とを用いて、情報データの送受信を行うデータ通信システムに適用可能なデータ通信方法において、上記1つ以上の受信機器の受信能力のうち、最も低い受信能力に従って上記情報データの送受信を実行させることを特徴としている。

【0021】本発明のデータ通信装置は、所定の機器に固有の固有情報と、1つ以上の受信機器との間の論理的な接続を示すコネクション情報とを用いて情報データの送受信を行う通信手段と、上記1つ以上の受信機器の受信能力のうち、最も低い受信能力に従って上記通信手段の動作を制御する制御手段とを具備することを特徴としている。

【0022】本発明の記憶媒体は、上記データ通信方法の手順をコンピュータに実行させるためのプログラムを格納したことを特徴としている。また、本発明の記憶媒体の他の特徴とするところは、上記各手段としてコンピュータを機能させるためのプログラムを格納したことを特徴としている。

【0023】

【作用】本発明は上記技術手段を有するので、コントローラノードによって設定されたコネクション識別データを使用してソース、デスティネーション間でデータ転送を行うことにより、バスリセットが発生して各ノードのノードIDが変化してもデータ転送を簡単な手順で再開することができる。また、複数のデスティネーションが異なるサイズのデータバッファを持つ場合には、コントロ

ーラノードがその中の最小のバッファサイズをソースに通知することにより、各デスティネーションの持つバッファサイズに関係なく、ソースは上記通知されたバッファサイズ以下のサイズ単位でデータ転送を行うことにより、通信を確実に実行することが可能となる。

【0024】

【発明の実施の形態】以下、図面を参照しながら、本発明のデータ通信システム、方法及び装置並びに記憶媒体の実施の形態について説明する。図1において、10はcomputerであり、12は演算処理装置(MPU)、14は第1の1394インターフェイス、16はキーボードなど第1の操作部、18は第1のデコーダ、20はCRTディスプレイなどの表示装置、22はハードディスク、24は第1のメモリであり本発明に係るcomputer10の内部メモリ、26はPCIバスなどのコンピュータ内部バスである。

【0025】また、28はVCRであり、30は撮像光学系、32はアナログ-デジタル(A/D)変換器、34はビデオ処理部、36は圧縮伸長回路、38は第1のメモリ、40は第2のメモリ、42は第1のデータセクタ、44は第2の1394インターフェイス、46は第1のメモリ制御回路、48は第2のメモリ制御回路、50はシステムコントローラ、52は第2の操作部、54はファインダ、56はD/A変換器、58は記録部である。

【0026】さらに、60はプリンタであり、62は第3の1394インターフェイス、64は第2のデータセクタ、66は第3の操作部、68はプリンタコントローラ、70は第2のデコーダ、72は第3のメモリ、74は画像処理部、76はドライバ、78はプリンタヘッドである。

【0027】computer10と、VCR28、及び、プリンタ60とは、第1から第3の1394インターフェイス14、44、62によって1394シリアルバスのノードを構成するとともに、該第1から第3の1394インターフェイス14、44、62を介して相互に接続されており、データの授受や、コマンドによるコントロール等が可能になっている。

【0028】本実施の形態では、例えば、computer10は、1394シリアルバス上における、画像信号送受信のコントローラとして動作する。本実施の形態に係るcomputer10においては、例えば、PCIバスなどのコンピュータ内部バス26によって、演算処理装置12と、1394インターフェイス14、キーボード16、デコーダ18、CRTディスプレイ20、ハードディスク22、内部メモリ24などの、内部の各デバイスとが相互に接続されている。

【0029】演算処理装置(MPU)12は、ハードディスク22に記録されているソフトウェアを実行するとともに、様々なデータを内部メモリ24に移動させる。また、演算処理装置(MPU)12は、PCIバス26によって接続されている各デバイスの、調停動作なども合わせて行なう。

【0030】1394インターフェイス14は、1394シリアルバス上に転送される画像信号を受信するとともに、ハードディスク22に記録されている画像信号や、内部メモリ

24に記憶される画像信号を送信する。また、1394インターフェイス14は、1394シリアルバス上に接続された他の機器に対するコマンドデータを送信する。また、1394インターフェイス14は、1394シリアルバス上に転送される信号を他の1394ノードに転送する。

【0031】操作者は、キーボード16などの操作部を通じて、MPU12に、ハードディスク22に記録されているソフトウェアを実行させる。該ソフトウェア等の情報は、CRTディスプレイなどの表示装置20によって、操作者に提示される。デコーダ18は、上記のソフトウェアを通じて、1394シリアルバス上から受信した画像信号をデコードする。デコードされた画像信号も、また、CRTディスプレイなどの表示装置20によって、操作者に提示される。

【0032】本実施の形態では、例えば、VCR28は、画像信号の入力装置として動作する。撮像光学系30から入力された映像の輝度信号(Y)と色差信号(C)は各々A/D変換器32にてデジタルデータに変換される。上記デジタルデータは、ビデオ処理部34にて多重化される。その後、圧縮伸長回路36にて該画像情報のデータ量を圧縮する。

【0033】一般に、YC独立に該圧縮処理回路を備えているが、ここでは説明の簡略化の為にYC時間分割での圧縮処理の例を示す。次に、上記画像データを伝送路誤りに強くする目的でシャフリング処理を施す。この処理の目的は連続的な符号誤りであるところのバーストエラーを、修整や補間の行い易い離散的な誤りであるところのランダムエラーに変換することである。

【0034】加えて、画像の画面内の粗密による情報量の発生の偏りを均一化する目的を重視する場合には上記圧縮処理の前に本処理工程を持ってくると、ランレングス等の可変長符号を用いた場合の都合が良い。

【0035】これを受けて、データ・シャフリングの復元のためのデータ識別(ID)情報を付加する。このID付加動作にて付加されたIDは、同時に記録しておいた上記システムのモード情報等と共に再生時の逆圧縮処理(情報量伸張処理)の際に補助情報として利用する。これらのデータの再生時の誤りを低減するためにエラー訂正(ECC)情報を付加する。

【0036】このような冗長信号の付加までを、映像と音声等の情報毎に対応する独立の記録エリア毎に処理する。上記のように、ID情報やECC情報が付加された画像信号は、記録部58により、磁気テープ等の記録媒体に記録されるとともに、後述する第1のメモリ38に一時的に記憶される。

【0037】一方、ビデオ処理部34にて多重化された画像データは、D/A変換器56によって、デジタル-アナログ変換され、電子ビューファインダ54で操作者により観察される。また、操作者は第2の操作部52を介して、様々な操作情報をシステムコントローラ50に送信し、シ

システムコントローラ50は、該操作情報によって、VCR全体を制御するようになっている。

【0038】また、ビデオ処理部34にて多重化された画像データは、第2のメモリ40に出力され、一時的に記憶される。前述した第1のメモリ38と、二のメモリ40とは、それぞれ、第1のメモリ制御回路46と、第2のメモリ制御回路48とを介し、システムコントローラ50により動作制御されている。

【0039】第1のデータセクタ42は、前述した第1のメモリ38と、第2のメモリ40からのデータを選択して、第2の1394インターフェイス44に受け渡す、あるいは、第2の1394インターフェイス44からのデータを選択して、第1のメモリ38と、第2のメモリ40とのどちらかに受け渡す。

【0040】上記動作により、VCR28における第2の1394インターフェイス44からは、圧縮された画像データと非圧縮の画像データとが、操作者により選択されて出力できるようになっている。

【0041】第2の1394インターフェイス44は、1394シリアルバスを通じて、VCR28を制御するためのコマンドデータを受信する。受信されたコマンドデータは、第1のデータセクタ42を通じて、システムコントローラ50に入力される。システムコントローラ50は、上記のコマンドデータに対するレスポンスデータを作成して、第1のデータセクタ42、及び、第2の1394インターフェイス44を通じ、1394シリアルバスに該データを送出する。

【0042】本実施の形態では、例えば、プリンタ60は、画像の印刷出力装置として動作する。第3の1394インターフェイス62は、1394シリアルバス上に転送される画像信号と、1394シリアルバスを通じて該プリンタ60を制御するためのコマンドデータとを受信する。また、第3の1394インターフェイス62は、該コマンドに対するレスポンスデータを送信する。

【0043】受信された画像データは、第2のデータセクタ64を通じて、第2のデコーダ70に入力される。第2のデコーダ70は、該入力された画像データをデコードして、画像処理部74に出力する。画像処理部74は、デコードされた画像データを第3のメモリ72に一時的に記憶する。

【0044】一方、受信されたコマンドデータは、第2のデータセクタ64を通じて、プリンタコントローラ68に入力される。プリンタコントローラ68は、該コマンドデータによりドライバ76による紙送り制御や、プリンタヘッド78の位置制御など、様々な印刷に関する制御を行なう。また、プリンタコントローラ68は、第3のメモリ72に一時的に記憶された画像データを、印刷データとして、プリンタヘッド78に送信し、印刷動作を行わせる。

【0045】上述したように、本実施の形態に係る、第1から第3の1394インターフェイス14,44,62は、それぞれ、1394シリアルバスのノードを構成する。第1の1394

インターフェイス14は、コントロールノード、または、コントローラとして動作し、第2の1394インターフェイス44は、画像データのソースノードとして動作し、第3の1394インターフェイス62は、デスティネーションノードとして動作する。

【0046】以下に、図2を用いて、本実施の形態のデータ転送動作の概要を説明する。図2において、200はコントローラ、202はソースノード、204はデスティネーションノード、206はソースノード内部のサブユニット、208は画像データ等のobject、210はデスティネーションノード内部の第1のメモリ空間、212は第1のコネクション、214はデスティネーションの第nのメモリ空間、216は第nのコネクション、である。

【0047】コントローラ200は、データ転送を行うソースノード202と、1つ以上のデスティネーションノード204との間の論理的な接続（以下、コネクション）を確立するためのコネクション識別データを設定するノードである。コントローラ200は、ソースノード202、及び、デスティネーションノード204と独立したノードであってもよいし、ソースノード、または、デスティネーションノードとコントローラとが同じであってもかまわない。

【0048】後者の場合、コントローラと同じノードである、ソースノード、または、デスティネーションノードと、コントローラとの間のトランザクションは、不要である。本実施の形態では、コントローラ200がソースノード202、及び、デスティネーションノード204とは別のノードに存在する場合の例を示す。

【0049】本実施の形態のデータ通信装置においては、複数のコネクションを確立することが可能である。コントローラはユーザが選択したデスティネーションノードとソースノードに同一のコネクション識別データをAsynchronousパケットを使用して設定する。

【0050】ソースノード202は、内部のサブユニット206から画像データ等のobject208を、例えば、第1のコネクション212を通じて、デスティネーションノード内部の第1のメモリ空間210にAsynchronousパケットを使用して書き込む。このときに使用するAsynchronousパケットは、例えば特定のノードIDを指定するデータを持たず、マルチキャスト用のIDを指定したパケットである。

【0051】このパケットにはコントローラによって設定されたコネクション識別データがデータヘッダ情報として書き込まれており、デスティネーションノードは第1のメモリ空間に書き込まれたデータのデータヘッダを解析し、コントローラによって設定されたコネクション識別情報を持つデータパケットである場合、データヘッダ情報を取り除いたデータを内部バッファに書き込んでいく。

【0052】デスティネーションノードが複数存在する

10

20

30

40

50

11

場合も、例えば、特定のノードIDを指定しないマルチキャスト用のIDを使用していることにより、同時に複数のノードに対してデータ転送を行うことができる。

【0053】データの転送が終了すると、コントローラ200はソースノード202、デスティネーションノード204に設定したコネクション識別データをクリアしてデータ転送を終了する。

【0054】次に、図3を用いて、Asynchronousパケットについて説明する。本実施の形態に係るAsynchronousパケットは、例えば、4 byte、(32 bits、以下クアドレットと称する)を単位とするデータパケットである。Asynchronousパケットは、送信先のノードIDを指定するパケットフォーマットと、Asynchronous Streamと呼ばれるチャンネル番号を指定するパケットフォーマットの2種類のフォーマットが存在する。本実施の形態では、オブジェクトデータを転送するパケットフォーマットとしていずれか一方のフォーマットを選択することができる。

【0055】図3(a)に示すパケットフォーマットは、ノードIDを指定するフォーマットである。最初の16 bitsはdestination_IDフィールドであり、該フィールドは受信先のノードIDを示す。次の6 bitsのフィールドは、トランザクション・ラベル(tl)フィールドであり、各トランザクション固有のタグである。

【0056】次の2 bitsのフィールドは、リトライ(rt)コードであり、パケットがリトライを試みるかどうかを指定する。次の4 bitsのフィールドは、トランザクションコード(tcode)である。tcodeは、パケットのフォーマットや、実行しなければならないトランザクションのタイプを指定する。

【0057】本実施の形態においては、例えば、この値が0001₂である、データブロックの書き込みリクエストのトランザクションを用いる。次の4 bitsのフィールドは、プライオリティ(pri)フィールドであり、優先順位を指定する。本実施の形態においては、Asynchronousパケットを用いているので、このフィールドの値は0000₂である。

【0058】次の16 bitsはsource_IDフィールドであり、送信側のノードIDを示す。次の48 bitsはdestination_offsetフィールドであり、パケットの受信先ノードアドレスの、下位48 bitsがこのフィールドによって指定される。

【0059】次の16 bitsはdata_lengthフィールドであり、後述するデータフィールドの長さを、バイト単位で示している。次の16 bitsはextended_tcodeフィールドであり、本実施の形態に用いられるデータブロックの書き込みリクエストトランザクションにおいては、この値は0000₁₆である。

【0060】次の32 bitsはheader_CRCフィールドであり、上述したdestination_IDフィールドからextend

12

ed_tcode フィールドまでを、パケットヘッダと称し、該ヘッダパケットのエラー検出に用いられる。

【0061】次のフィールドは可変長のデータフィールドであり、該データフィールドをパケットのペイロードと称する。本実施の形態においては、該データフィールドがクアドレットの倍数でない場合、クアドレットに満たないビットには0が詰められる。次の32 bitsのフィールドはdata_CRCフィールドであり、上記のheader_CRCフィールドと同様に、該データフィールドのエラー検出に用いられる。

【0062】図3(b)に示すパケットフォーマットは、チャンネル番号を指定するAsynchronous streamフォーマットである。最初の16 bitsはdata_lengthフィールドであり、後述するデータフィールドの長さを、バイト単位で示している。

【0063】次の2 bitsはtagフィールドであり、この値は00₂である。次の6 bitsはchannelフィールドであり、このパケットのチャンネル番号を表す。受信ノードはこのチャンネル番号を用いてパケットを識別する。次の4 bitsはトランザクションコード(tcode)である。Asynchronous streamパケットでは、A1₆である。

【0064】次の4 bitsはSynchronization code(sy)フィールドであり、このパケットを使用するアプリケーションによって、この値は決定される。次の32 bitsはheader_CRCフィールドであり、上述したdata_lengthフィールドからsyフィールドフィールドまでを、パケットヘッダと称し、該ヘッダパケットのエラー検出に用いられる。

【0065】次のフィールドは可変長のデータフィールドであり、該データフィールドをパケットのペイロードと称する。本実施の形態においては、該データフィールドがクアドレットの倍数でない場合、クアドレットに満たないビットには0が詰められる。

【0066】次の32 bitsのフィールドはdata_CRCフィールドであり、上記のheader_CRCフィールドと同様に、該データフィールドのエラー検出に用いられる。次に、図4を用いて、上述した、コントローラ200、ソースノード202、デスティネーションノード204の間で行われるAsynchronousトランザクションについて説明する。本実施の形態では1つのソースノードと1つのデスティネーションノードの間にコネクションを設定した場合を説明する。

【0067】コントローラはユーザーが選択したデスティネーションノードに対して、コネクションを設定するためのSET DESTINATION コマンドパケットを送信する。このパケットにはコントローラノードのROM内に書き込まれているnode_vendor_id, chip_id_hi, chip_id_loのデータとこのコネクションに割り当てられたconnection_idが書き込まれている。

【0068】1394ノードが持つnode_vendor_id, chip

10

20

30

40

50

__id_hi,chip __id_loを組み合わせた64ビットのデータはEUI-64(Extended Unique Identifier, 64bits)と呼ばれ、そのノード固有のデータであり、1つの通信システム内に同じEUI-64を持つ1394ノードは存在しない。本実施の形態ではEUI-64とconnection_idを組み合わせたデータをコネクション識別データとして使用する。

【0069】バス上に複数のコントローラが存在する場合でも各コントローラのEUI-64とconnection_idの組み合わせをコネクション識別データとすることにより、各コントローラは各コントローラがコネクションに割り当てたconnection_idのみを管理することでバス上で一意に定まるコネクション識別データを管理することができる。

【0070】図5に、上記コマンドパケットのフォーマット例を示す。

値	コマンドタイプ	意味
0	Control	制御コマンド
1	Status	機器の状態問い合わせ
2	Inquiry	該当コマンドのサポート状況問い合わせ
3	Notify	機器の状態変化の確認

【0073】図7に、上記SET DESTINATION コマンドパケットに使用されるopcode,operandの例を示す。opcodeフィールドには、デスティネーションに対してコネクション識別データを設定するためのパケットであることを示すコードがセットされる。

【0074】operand[0]～operand[2],operand[3],operand[4]～operand[7]のフィールドにはそれぞれコントローラノードのROM 内に書き込まれているnode_vendor_id,chip __id_hi,chip __id_loのデータがセットされる。operand[8]にはコントローラによって管理されるconnection_id 値がセットされる。コントローラからデスティネーションに対してこのパケットを送信する場合は残りのoperand のフィールドにはダミーデータをセットする。

【0075】上記SET DESTINATION コマンドパケットを※

値	レスポンスタイプ	意味
8	Not Implemented	該当コマンドはサポートされていない
9	Accepted	コマンドを受け付けた
A ₁₆	Rejected	コマンドを拒否した
F ₁₆	Interim	後でレスポンスを返す

【0078】デスティネーションがデータランザクションのコネクションを設定することが可能な場合は、responseフィールドにInterim を設定し（ソースノードとのデータランザクションが終了した時点でAcceptedのレスポンスを再度コントローラノードに送信する）、不可能な場合はRejectedを設定する。opcode,operandフィールドは図7に示すフォーマットで、opcodeにはSET DESTINATION

* ット例を示す。ここに示すフォーマットのデータは図3(a) のデータフィールドにAsynchronousパケットのペイロードとしてセットされてデスティネーションノードに送信される。

【0071】ctype フィールドはコマンドの種類を示す物であり、表1に示すコマンドタイプを指定する。SET DESTINATION コマンドの場合はControl を指定する。subunit __type,subunit ID フィールドは、指定したノード内のどのユニットに対してのコマンドなのかを示すフィールドである。opcode,operandフィールドは実際のコマンドを内容に指定するフィールドである。

【0072】

【表1】

※受け取ったデスティネーションノードは、コントローラノードにSET DESTINATION レスポンスパケットを送信する。図6に、このレスポンスパケットのフォーマット例を示す。ここに示すフォーマットのデータは図3のデータフィールドにAsynchronousパケットのペイロードとしてセットされてコントローラノードに送信される。

【0076】responseフィールドは、レスポンスの種類を示す物であり、表2に示すレスポンスタイプを指定する。subunit __type,subunit ID フィールドは、ノード内のどのユニットからのレスポンスなのかを示すフィールドである。opcode,operandフィールドはレスポンスデータを指定するフィールドである。

【0077】

【表2】

★SET DESTINATION レスポンスパケットであることを示すコードを、node_vendor_id,chip __id_hi,chip __id_lo,connection __idフィールドはコントローラから指定されたデータを設定する。

【0079】コネクション設定が可能な場合はmax __rec フィールドにIEEE1394で規定されるデスティネーションノードのmax __rec 値(Asynchronous の1パケットで

受信できるデータサイズを示すデータである)を設定し、buffer_size フィールドにデータ受信に使用できるバッファサイズを設定する。

【0080】コネクション設定が不可能な場合、max_rec,buffer_size フィールドはコントローラからのコントロールコマンドパケットに設定されていたダミーデータと同じデータを設定する。status_infoフィールド*

Value	meaning
00 ₁₆	Success
01 ₁₆	Aborted
11 ₁₆	Busy
12 ₁₆	Serial Bus Error
13 ₁₆	Hardware trouble

【0082】デスティネーションノードからInterim のSET DESTINATION レスポンスパケットを受信した後、コントローラノードはユーザが選択したソースノードに対して、コネクションを設定するためのSET SOURCEコマンドパケットを送信する。

【0083】このパケットには上記デスティネーションノードに設定したnode_vendor_id,chip_id_hi,chip_id_lo,connection_idが書き込まれている。このコマンドパケットのフォーマットも図5に示すものでctype フィールドにはControlが設定される。また上記フォーマットのデータは図3(a)のデータフィールドにAsynchronousパケットのペイロードとしてセットされてソースノードに送信される。

【0084】図8に上記SET SOURCEコマンドパケットに格納するデータの一例を示す。opcodeフィールドには、ソースノードに対してコネクション識別データを設定するためのパケットであることを示すコードがセットされる。

【0085】operand[0]~operand[2],operand[3],operand[4]~operand[7],operand[8]のフィールドには上記デスティネーションノードに設定したnode_vendor_id,chip_id_hi,chip_id_lo,connection_idのデータがセットされる。

【0086】operand[9]のmax_rec フィールド及びoperand[10]~operand[12]のbuffer_size には、デスティネーションノードからのSET DESTINATION レスポンスパケットにセットされていたmax_rec,buffer_size のデータがセットされる。operand[13]のnumber_of_destinationsフィールドにはデスティネーションの数がセットされる。

【0087】図4に示すフローでは、1対1のトランザクションなので、この場合は1がセットされる。operand[14]のstatus_infoにはダミーデータがセットされる。上記SET SOURCEコマンドパケットを受け取ったソースノードは、コントローラノードにSET SOURCEレスポンス※50

*にはコマンドの実行状況を設定する。表3にSET DESTINATION レスポンスパケットのstatus値の例を示す。コネクション設定可能な場合はSuccess を、不可能な場合はBusyを設定する。

【0081】

【表3】

※パケットを送信する。SET SOURCEレスポンスパケットのフォーマットは図6に示すフォーマットである。また、上記フォーマットのデータは図3(a)のデータフィールドにAsynchronousパケットのペイロードとしてセットされてコントローラノードに送信される。

【0088】ソースノードがデータトランザクションのコネクションを設定することが可能な場合は、response フィールドにAcceptedを設定し、不可能な場合はRejectedを設定する。opcode,operandフィールドは図8に示すデータが格納される。opcodeにはSET SOURCEレスポンスパケットであることを示すコードを、node_vendor_id,chip_id_hi,chip_id_lo,connection_idフィールドはコントローラから指定されたデータを設定する。

【0089】データトランザクションのコネクションを設定できる場合、パケットからコネクション識別データを取りだしてノード内部のバッファに記憶する。max_rec,buffer_sizeフィールドはSET SOURCEコマンドパケットにセットされていたデータを設定する。

【0090】本実施の形態では、ソースノードと複数のデスティネーションノード間でのコネクションを設定できるが、Asynchronous受信用のFIFO容量等の関係で、各ソースノードは同時に接続できるデスティネーションノードの数が制限される。

【0091】SET SOURCEコマンドパケットのnumber_of_destinationsフィールドの値を調べて、接続可能な数値以下の場合はその値をSET SOURCEレスポンスパケットのnumber_of_destinationsフィールドにセットする。接続可能な数値を超える場合は、接続可能なノード数の値をセットする。

【0092】status_infoフィールドはコマンドの実行状況を設定する。表4にSET SOURCEレスポンスパケットのstatus値の例を示す。コネクション設定可能な場合で、デスティネーションノード数が接続可能なノード数以下の場合はSuccess を、接続可能なノード数を超えて

いる場合はToo many destinations を設定する。コネクション設定不可能な場合はBusyを設定する。 * 【0093】 * 【表4】

Value	meaning
00 ₁₆	Success
02 ₁₆	Too many destinations
11 ₁₆	Busy
12 ₁₆	Serial Bus Error
13 ₁₆	Hardware trouble

【0094】ソースノードからAcceptedのSET SOURCEレスポンスパケットを受信した後、コントローラノードはSET SOURCEレスポンスパケットのstatus_infoフィールドを調べてSuccessであることを確認すると、ソースノードに対してOBJECT SENDコマンドパケットを送信する。

【0095】ソースノードでは本実施の形態で示されない何らかの手段によって、あらかじめ送信を行うデータが選択されているものとする。このパケットには上記ソースノードに設定したnode_vendor_id, chip_id_hi, chip_id_lo, connection_idが書き込まれている。

【0096】このコマンドパケットのフォーマットも図5に示すものでctype フィールドにはControl が設定される。また、上記フォーマットのデータは図3(a)のデータフィールドにAsynchronousパケットのペイロードとしてセットされてソースノードに送信される。

【0097】図9に、上記OBJECT SEND コマンドパケッ※

subfunction	Value	Action
send	00 ₁₆	Perform a normal operation
abort	02 ₁₆	Perform a "abort command" operation

【0100】上記OBJECT SEND コマンドパケットを受け取ったソースノードは、コントローラノードにOBJECT SEND レスポンスパケットを送信する。OBJECT SEND レスポンスパケットのフォーマットも図6に示すフォーマットである。また上記フォーマットのデータは図3(a)のデータフィールドにAsynchronousパケットのペイロードとしてセットされてコントローラノードに送信される。

【0101】ソースノードはOBJECT SEND コマンドパケットに設定されているコネクション識別データを調べて、自ノードに設定されているデータと一致する場合でデータ送信開始が可能な場合はresponseフィールドにInterimを設定し（デスティネーションノードとのデータトランザクションが終了した時点でAcceptedのレスポンスを再度コントローラノードに送信する）、そうでない場合はRejectedを設定する。★

※トに格納されるデータの一例を示す。opcodeフィールドには、ソースノードに対して選択されているオブジェクトデータの送信開始を指示するパケットであることを示すコードがセットされる。

【0098】operand[0]～operand[2], operand[3], operand[4]～operand[7], operand[8] のフィールドには上記ソースノードに設定したnode_vendor_id, chip_id_hi, chip_id_lo, connection_idのデータ、すなわち、コントローラのEUI-64がセットされる。operand[9]のsubfunction フィールドにはこのコマンドパケットで指示される実際の動作を表すコードがセットされる。このコードは表5に示すようなコードで、データ送信の開始を指示する場合はsendがセットされる。operand[10]のstatus_infoにはダミーデータがセットされる。

【0099】

【表5】

★【0102】opcode, operandフィールドは、図9に示すデータが格納される。opcodeにはOBJECT SEND レスポンスパケットであることを示すコードを、node_vendor_id, chip_id_hi, chip_id_lo, connection_id, subfunctionフィールドにはコントローラからのOBJECT SEND コマンドパケットに設定されていたデータを設定する。

【0103】status_infoフィールドはコマンドの実行状況を設定する。表6にOBJECT SEND レスポンスパケットのstatus値の例を示す。データ送信開始が可能な場合はSuccessを設定し、そうでない場合は送信開始が不可能な理由に応じたデータを設定する。

【0104】

【表6】

19

20

Value	meaning
00 ₁₆	Success
01 ₁₆	Aborted
11 ₁₆	Busy
12 ₁₆	Serial Bus Error
13 ₁₆	Hardware trouble
14 ₁₆	Unknown connection id

【0105】以上のAsynchronousトランザクションをコントローラノード、デスティネーションノード、ソースノードの間で行うことにより、コネクションの設定が完了し、オブジェクトデータのトランザクションが開始される。上記SET DESTINATION, SET SOURCE, OBJECT SENDパッケージはそれぞれ図3(a)のdestination __IDフィールドに受信ノードのノードIDを指定したパッケージを使用して各ノードから送信が行われる。

【0106】コントローラノードはconnection_id管理用のテーブルを持ち、コネクションに使用されているconnection_idを管理している。例えば、図19に示すような、connection_idに対応したフラグレジスタ及びmax __rec, buffer_sizeを記憶するためのバッファを持ち、使用中のconnection_idのflagレジスタに使用中のビットをセットして、そのconnection_idに対するmax __rec 値及びbuffer_size値を保持する。このflagレジスタを調べて、使用されていないconnection_idを割り当てることにより、同時に複数のコネクションを設定することができる。

【0107】次に、ソースノードとデスティネーションノードの間で行われるオブジェクトデータのトランザクションについて説明する。図10に示すように、ソースノードでは送信オブジェクトとして128KBの静止画像データが選択されており、デスティネーションノードは32KBのデータ受信用のバッファを持つものとする。また、デスティネーションノードは512バイトまでのAsynchronousパッケージを受信できるものとする。

【0108】この場合、デスティネーションノードは上述のSET DESTINATION レスポンスでmax __rec フィールドに512バイト、buffer_sizeフィールドに32KBを示すデータをセットしてコントローラにレスポンスを送信し、コントローラはこのデータをSET SOURCEコマンドを使用してソースノードに通知する。

【0109】ソースノードは、選択されているオブジェクトをmax __rec 値を超えないsegment データに分割して（本実施の形態では256byteのsegment データ）各セグメントをデスティネーションに送信する。図10では、オブジェクトデータを等しいサイズのsegment に分*

*割している。

【0110】図4のフローに示すように、セグメント単位でソースノードはオブジェクトデータを送信する。この送信にはAsynchronousトランザクションを使用するが、図3(a)のAsynchronousパッケージを使用する場合、destination __idフィールドには特定のデスティネーションノードを示すノードIDではなく、マルチキャスト用のIDが設定される。

【0111】本実施の形態では、例えばFFDF₁₆が設定され、オブジェクトデータの通信を行うために、destination __idがこの値にセットされているパッケージをソースノードとデスティネーションノードとの間で送受信する。また、図3(b)のフォーマットのAsynchronous stream パッケージフォーマットを使用する場合、特定のデスティネーションノードを示すノードIDではなく、所定のノードにより管理されたチャンネル番号がchannel フィールドにセットされる。

【0112】この場合、コントローラがオブジェクトデータの転送前に、上記チャンネル番号をソースノードとデスティネーションに通知する。ソースノードとデスティネーションとは、このチャンネル番号がセットされたパッケージを用いてオブジェクトデータの通信を行う。

【0113】以下では、図3(a)のパッケージフォーマットを使用した例を説明する。図11に、Segmentに分割されたオブジェクトデータの送信パッケージフォーマット例を示す。このフォーマットのデータは図3(a)に示すdestination __idフィールドにマルチキャスト用のIDが設定されたAsynchronousパッケージのData fieldにセットされてソースノードから送信される。

【0114】node __vendor __id, chip __id __hi, chip __id __lo, connection __idフィールドには、コントローラから設定されたコネクション識別データが設定される。control __flags は、このデータパッケージのタイプを示す情報がセットされる。表7にソースノードで設定するcontrol __flags の値の例を示す。

【0115】

【表7】

21

22

Value	meaning
00 ₁₆	Normal data
01 ₁₆	Buffer end
02 ₁₆	Object end

【0116】sequence_numberフィールドには、このパケットで送信されるsegmentデータの通し番号がセットされる。Segmented object dataの部分に、segment分割されたオブジェクトのデータがセットされる。また、図3(b)に示すパケットを使用する場合も同様に、図11に示すフォーマットがdata fieldにセットされる。

【0117】まず、ソースノードの動作について説明する。ソースノードは、送信したsegmentデータのトータルサイズをカウントし、コントローラからSET SOURCEコマンドで通知されたデスティネーションのバッファサイズに達するまで順次データの送信を行う。

【0118】この場合の各パケットのcontrol_flagsフィールドにはNormal dataがセットされる。送信データのトータルサイズがデスティネーションのバッファサイズを超えない適当なサイズに達した場合、そのパケットのcontrol_flagsフィールドにBuffer endをセットしてデータを送信する。

【0119】図10の例では、トータルサイズがデスティネーションのバッファサイズと等しくなるsequence_n*

Value	meaning
10 ₁₆	Receive success
11 ₁₆	Resend request

【0123】control_flagsの値がReceive successの場合は正しく受信できたデータのシーケンス番号が、Resend requestの場合は再送を希望するデータのシーケンス番号がセットされる。

【0124】ソースノードは、destination_idにマルチキャスト用のIDが設定されているAsynchronousパケットを受信して、ペイロード中のコネクション識別データを調べて、自ノードに設定されているデータと一致する場合にcontrol_flagsの値を調べる。この値がReceive successの場合、送信segmentデータのトータルカウントをクリアして次のsegmentからデータ送信を開始する。

【0125】図10の例では、sequence numberが255, 383のsegmentデータの送信時にcontrol_flagsをBuffer endにセットして、同様にデスティネーションからのレスポンスを受け取った後にデータ送信を再開する。

【0126】オブジェクトの最終データの送信時にはcontrol_flagsをObject endに設定してソースノードはデータを送信する。Buffer endの場合と同様にソースノードはレシーブレスポンスを待機し、デスティネーション

*umberが127のsegmentの送信時に、control_flagsをBuffer_endにセットする。その後、デスティネーションノードからレシーブレスポンスパケットが送信されるのを待機する。

【0120】図12にレシーブレスポンスのパケットフォーマット例を示す。このフォーマットのデータは、図3(a)に示すdestination_idフィールドにマルチキャスト用のIDが設定されたAsynchronousパケットのData fieldにセットされてデスティネーションノードから送信される。

【0121】node_vendor_id, chip_id_hi, chip_id_lo, connection_idフィールドには、コントローラから設定されたコネクション識別データが設定される。control_flagsは、このデータパケットのタイプを示す情報がセットされる。表8にデスティネーションノードで設定するcontrol_flagsの値の例を示す。

【0122】

【表8】

※ンからのレシーブレスポンスのcontrol_flagsがReceive successの場合、コントローラノードに対して、OBJECT_SENDレスポンスパケットを送信する。

【0127】このAsynchronousパケットのdestination_idにはコントローラノードのノードIDを設定して送信する。図6のresponseフィールドにはAcceptedをセットし、図9のopcode, operand[0]～operand[9]にはOBJECT_SENDコマンドを受け取った直後にコントローラに対して送信したInterimのOBJECT_SENDレスポンスと同一のデータを設定する。

【0128】operand[10]のstatus_infoフィールドにはデータ送信の終了状態を示すコードをセットする。正常に終了した場合はSuccessがセットされる。次に、デスティネーションノードの動作について説明する。

【0129】デスティネーションノードは、destination_idフィールドにマルチキャスト用のIDが設定されているAsynchronousパケットを受信するとデータフィールド中のコネクション識別データを調べて、自ノードに設定されているデータと一致した場合、受信バッファの先頭アドレスから順次segmentデータを書き込んでい

く。また、このときsequence number フィールドを調べ、欠落データの検出を行うことができる。

【0130】データパケットのcontrol _flags を調べて、この値がNormal data の場合は次のデータパケットが送信されるのを待つ。control _flags の値がBuffer endの場合、受信用バッファに書き込んだデータを別のバッファ（ハードディスク等）にコピーして、バッファをクリアした後に、図12に示すフォーマットのレスポンスパケットをマルチキャスト用IDを使用して送信する。

【0131】このとき、control _flags にはReceive success を、sequence number にはBuffer endが設定されたデータパケットのsequence number をセットする。図10の例では127 がセットされる。

【0132】レスポンスを送信した後、デスティネーションノードはソースノードからデータの送信が再開されるのを待機し、データを受信すると再び受信用バッファの先頭から順次segment データを書き込んでいく。図10の例の場合、sequence number が255,383 のsegment データを受信すると、同様にレスポンスを送信する。

【0133】control _flags の値がObject endのsegment データを受信すると、デスティネーションノードは同様にレスポンスを送信する。Object endに対するレスポンスをマルチキャスト用IDを使用して送信した後、デスティネーションノードはコントローラノードに対してSET DESTINATION レスポンスパケットを送信する。

【0134】このAsynchronousパケットのdestination _idにはコントローラノードのノードIDを設定して送信する。図6のresponseフィールドにはAcceptedをセットし、図7のopcode, operand[0] ~operand[12] にはSET DESTINATION コマンドを受け取った直後にコントローラに対して送信したInterim のSET DESTINATION レスポンスと同一のデータを設定する。

【0135】operand[13] のstatus_info フィールドにはデータ送信の終了状態を示すコードをセットする。正常に終了した場合はSuccess がセットされる。上述の手順でソースノードとデスティネーションノードの間のオブジェクト転送が完了する。

【0136】ソースノードとデスティネーションノードからそれぞれOBJECT SEND, SET DESTINATION のレスポンスパケットを受信した後、コントローラノードはソースノード、デスティネーションノードに対してコネクションを解除するためのCLEAR CONNECTIONコマンドパケットを送信する。

【0137】このコマンドパケットのフォーマットは図5に示すものでctype フィールドにはControl が設定される。また上記フォーマットのデータは図3(a) のデータフィールドにAsynchronousパケットのペイロードとし

てセットされてソースノード、デスティネーションノードに送信される。

【0138】図13に、上記CLEAR CONNECTIONコマンドパケットのopcode, operandフィールドに格納するデータの一例を示す。opcodeフィールドには、コネクションの解除パケットであることを示すコードがセットされる。operand[0]~operand[2], operand[3], operand[4]~operand[7], operand[8] のフィールドには解除を行うコネクション識別データに使用されているnode_vendor_id, chip _id_hi, chip _id_lo, connection _idのデータがセットされる。

【0139】operand[9]のstatus_infoにはコネクション解除の理由を示すコードがセットされる。表9にデスティネーションノードに対してCLEAR CONNECTIONコマンドを送信する場合のstatus_infoの値の例を示す。

【0140】

【表9】

Value	meaning
00 ₁₆	Transfer is over
01 ₁₆	Transfer is aborted
12 ₁₆	Serial Bus Error
20 ₁₆	Source busy
21 ₁₆	Too many Destinations
22 ₁₆	Source error
FF ₁₆	No information

【0141】また、表10にソースノードに対してCLEAR CONNECTIONコマンドを送信する場合のstatus_infoの値の例を示す。

【0142】

【表10】

Value	meaning
00 ₁₆	Transfer is over
01 ₁₆	Transfer is aborted
12 ₁₆	Serial Bus Error
20 ₁₆	Destination busy
22 ₁₆	Destination error
FF ₁₆	No information

【0143】正常にデータ転送が終了している場合はソースノード、デスティネーションノードのどちらに対してもstatus_infoフィールドにはTransfer is overがセットされる。

【0144】上記CLEAR CONNECTIONコマンドを受信したソースノード及びデスティネーションノードはコネクション識別データがそれぞれ自ノードに設定されているデ

25

ータと一致する場合、それぞれのノードが内部バッファに記憶していたコネクション識別データをクリアして、コントローラノードに対してCLEAR CONNECTIONレスポンスパケットを送信する。

【0145】このレスポンスパケットのフォーマットは図6に示すもので正常にコネクションの解除が行われた場合はresponseフィールドにはAcceptedが設定される。また上記フォーマットのデータは図3(a)のデータフィールドにAsynchronousパケットのペイロードとしてセットされてコントローラノードに送信される。

【0146】上記CLEAR CONNECTIONレスポンスパケットのopcode,operandフィールドに格納されるデータの一例を図13に示す。opcodeにはCLEAR CONNECTIONレスポンスパケットであることを示すコードを、node_vendor_id,chip_id_hi,chip_id_lo,connection_id,subfunctionフィールドには解除したコネクション識別データに使用していたnode_vendor_id,chip_id_hi,chip_id_lo,connection_idを設定する。

【0147】status_infoフィールドはコマンドの実行状況を設定する。表11にCLEAR CONNECTIONレスポンスパケットのstatus値の例を示す。正常にコネクションを解除した場合はSuccessをセットする。

【0148】

【表11】

Return value	meaning
00 ₁₆	Success
11 ₁₆	Busy
12 ₁₆	Serial Bus Error
14 ₁₆	Unknown connection id

【0149】ソースノード、デスティネーションノードからAcceptedのCLEAR CONNECTIONレスポンスパケットを受信したコントローラノードは、コネクションを解除したconnection_idフラグレジスタの使用中のビットをクリアする。以上の手順で、ソースノードとデスティネーションノードの間に設定された論理的なコネクションが解除され、全てのトランザクションが終了する。

【0150】このように、コントローラが同一のコネクション識別データをデスティネーションノード及びソースノードに設定することによって、ソースノードとデスティネーションノード間に論理的なコネクションを設定することができ、データ転送処理はコントローラが介在しない、ソース・デスティネーション間のみのトランザクションで行うことができる。

【0151】(第2の実施の形態)次に、図14を用いて第2の実施の形態を説明する。本実施の形態では図2に示したコントローラ200、ソースノード202、デスティネーションノード204との間でコネクションを設定してデータ転送を行う。

26

【0152】コントローラはユーザーが選択したデスティネーションノードに対して、コネクションを設定するためのSET DESTINATION コマンドパケットを送信する。このパケットにはコントローラノードのROM内に書き込まれているnode_vendor_id,chip_id_hi,chip_id_loのデータとこのコネクションに割り当てられたconnection_idが書き込まれている。

【0153】上記SET DESTINATION コマンドパケットのフォーマットは図5に示すものでctype フィールドにはControl が設定される。また上記フォーマットのデータは図3(a)のデータフィールドにAsynchronousパケットのペイロードとしてセットされてデスティネーションノードに送信される。

【0154】図15に、上記SET DESTINATION コマンドパケットに使用されるopcode,operandの例を示す。opcodeフィールドには、デスティネーションに対してコネクション識別データを設定するためのパケットであることを示すコードがセットされる。

【0155】operand(0)～operand(2),operand(3),operand(4)～operand(7)のフィールドにはそれぞれコントローラノードのROM内に書き込まれているnode_vendor_id,chip_id_hi,chip_id_loのデータがセットされる。

【0156】operand(8)にはコントローラによって管理されるconnection_id値がセットされる。コントローラからデスティネーションに対してこのパケットを送信する場合は残りのoperandのフィールドにはダミーデータをセットする。

【0157】上記SET DESTINATION コマンドパケットを受け取ったデスティネーションノードは、コントローラノードにSET DESTINATION レスポンスパケットを送信する。上記SET DESTINATION レスポンスパケットのフォーマットは図6に示すもので、図3(a)のデータフィールドにAsynchronousパケットのペイロードとしてセットされてコントローラノードに送信される。

【0158】デスティネーションがデータトランザクションのコネクションを設定することが可能な場合はresponseフィールドにInterimを設定し(ソースノードとのデータトランザクションが終了した時点でAcceptedのレスポンスを再度コントローラノードに送信する)、不可能な場合はRejectedを設定する。

【0159】opcode,operandフィールドは図15に示すフォーマットで、opcodeにはSET DESTINATION レスポンスパケットであることを示すコードを、node_vendor_id,chip_id_hi,chip_id_lo,connection_idフィールドはコントローラから指定されたデータを設定する。destination_node_vendor_id,destination_chip_id_hi,destination_chip_id_loフィールドにはデスティネーションノードのROMに書き込まれているnode_vendor_id,chip_id_hi,chip_id_loを設定す

る。

【0160】コネクション設定が可能な場合はmax_rec フィールドにIEEE1394で規定されるデスティネーションノードのmax_rec 値(Asynchronous の1パケットで受信できるデータサイズを示すデータである)を設定し、buffer_size フィールドにデータ受信に使用できるバッファサイズを設定する。

【0161】コネクション設定が不可能な場合、max_rec,buffer_size フィールドはコントローラからのコントロールコマンドパケットに設定されていたダミーデータと同じデータを設定する。status_infoフィールドにはコマンドの実行状況を設定する。SET_DESTINATION レスポンスパケットのstatus値の例は表3に示した通りである。コネクション設定可能な場合はSuccess を、不可能な場合はBusyを設定する。

【0162】デスティネーションノードからInterim のSET_DESTINATION レスポンスパケットを受信した後、コントローラノードはユーザが選択したソースノードに対して、コネクションを設定するためのSET_SOURCEコマンドパケットを送信する。このパケットには上記デスティネーションノードに設定したnode_vendor_id,chip_id_hi,chip_id_lo,connection_idが書き込まれている。

【0163】このコマンドパケットのフォーマットは図5に示すもので、ctype フィールドにはControl が設定される。また上記フォーマットのデータは図3(a)のデータフィールドにAsynchronousパケットのペイロードとしてセットされてソースノードに送信される。

【0164】図16に上記SET_SOURCEコマンドパケットの一例を示す。opcodeフィールドには、ソースノードに対してコネクション識別データを設定するためのパケットであることを示すコードがセットされる。

【0165】node_vendor_id,chip_id_hi,chip_id_lo,connection_idフィールドには上記デスティネーションノードに設定したnode_vendor_id,chip_id_hi,chip_id_lo,connection_idのデータがセットされる。max_rec フィールド及びbuffer_size フィールドには、デスティネーションノードからのSET_DESTINATION レスポンスパケットにセットされていたmax_rec,buffer_sizeのデータがセットされる。number_of_destinationsフィールドにはデスティネーションの数がセットされる。

【0166】図14に示すフローでは1対1のトランザクションなので、この場合は1がセットされる。source_node_vendor_id,source_chip_id_hi,source_chip_id_lo,status_infoにはダミーデータがセットされる。

【0167】上記SET_SOURCEコマンドパケットを受け取ったソースノードは、コントローラノードにSET_SOURCEレスポンスパケットを送信する。SET_SOURCEレスポンス

パケットのフォーマットは図6に示すフォーマットである。また上記フォーマットのデータは図3(a)のデータフィールドにAsynchronousパケットのペイロードとしてセットされてコントローラノードに送信される。

【0168】ソースノードがデータトランザクションのコネクションを設定することが可能な場合はresponseフィールドにAcceptedを設定し、不可能な場合はRejectedを設定する。opcode,operandフィールドは図16に示すフォーマットで、opcodeにはSET_SOURCEレスポンスパケットであることを示すコードを、node_vendor_id,chip_id_hi,chip_id_lo,connection_idフィールドはコントローラから指定されたデータを設定する。

【0169】データトランザクションのコネクションを設定できる場合、パケットからコネクション識別データを取りだしてノード内部のバッファに記憶する。source_node_vendor_id,source_chip_id_hi,source_chip_id_loフィールドにはソースノードのROM に書き込まれているnode_vendor_id,chip_id_hi,chip_id_loを設定する。

【0170】max_rec,buffer_sizeフィールドはSET_SOURCEコマンドパケットにセットされていたデータを設定する。本実施の形態ではソースノードと複数のデスティネーションノード間でのコネクションを設定できるが、Asynchronous受信用のFIFO容量等の関係で、各ソースノードは同時に接続できるデスティネーションノードの数が制限される。

【0171】SET_SOURCEコマンドパケットのnumber_of_destinationsフィールドの値を調べて、接続可能な数値以下の場合はその値をSET_SOURCEレスポンスパケットのnumber_of_destinationsフィールドにセットする。接続可能な数値を超える場合は、接続可能なノード数の値をセットする。status_infoフィールドはコマンドの実行状況を設定する。

【0172】SET_SOURCEレスポンスパケットのstatus値の例は表4に示す通りである。コネクション設定可能な場合で、デスティネーションノード数が接続可能なノード数以下の場合にはSuccess を、接続可能なノード数を超えている場合はToo many destinations を設定する。コネクション設定不可能な場合はBusyを設定する。

【0173】ソースノードからAcceptedのSET_SOURCEレスポンスパケットを受信した後、コントローラノードはSET_SOURCEレスポンスパケットのstatus_infoフィールドを調べてSuccess であることを確認すると、ソースノードに対してOBJECT_SENDコマンドパケットを送信する。OBJECT_SEND コマンドの動作については第1の実施の形態と同様であるので説明を省略する。

【0174】以上のAsynchronousトランザクションをコントローラノード、デスティネーションノード、ソースノードの間で行うことにより、コネクションの設定が完了し、オブジェクトデータのトランザクションが開始さ

10

20

30

40

50

れる。上記SET DESTINATION, SET SOURCE, OBJECT SEND パケットはそれぞれ図3(a)のdestination __IDフィールドに受信ノードのノードIDを指定したパケットを使用して各ノードから送信が行われる。

【0175】コントローラノードはconnection_id管理用のテーブルを持ち、コネクションに使用されているconnection_idを管理している。たとえば図20に示すような、connection_idに対応したflagレジスタ及びmax __rec, buffer_size, デスティネーションノード数, デスティネーションノード・ソースノードのEUI-64の値を記憶するためのデータバッファを持ち、使用中のconnection_idのflagレジスタに使用中のビットをセットして、そのconnection_idに対する上述の各データを保持する。このフラグレジスタを調べて、使用されていないconnection_idを割り当てることにより、同時に複数のコネクションを設定することができる。

【0176】OBJECT SEND コマンドに従って、第1の実施の形態と同様にオブジェクトデータの転送が開始される。図14中のData Segment __m+1のデータをデスティネーションノードが受信した後にバスリセットが発生した場合、バス上の各ノードはIEEE1394の規格に定められた手順でバスの再構築を行う。これにより、ソースノードとデスティネーションノードのノードIDは再設定される。

【0177】バスの再構築が終了すると、デスティネーションノードはマルチキャストIDをdestination __idフィールドに設定したAsynchronousパケット(図3(a)に示す)を使用して、図12に示すフォーマットのレスポンスパケットを送信する。また、オブジェクトデータのトランザクションを図3(b)に示すパケットフォーマットにて行っている場合、デスティネーションは、所定のチャンネル番号をchannel フィールドに格納したパケットを用いてレスポンスパケットを送信する。

【0178】node_vendor_id, chip __id_hi, chip __id_lo, connection __idフィールドには、それぞれバスリセット前にコントローラノードによって設定されたnode_vendor_id, chip __id_hi, chip __id_lo, connection __idの値をセットする。control __flags にはResend requestを、sequence number にはバスリセット前に受信できたsegment のシーケンス番号に1を加えたm+2をセットする。

【0179】バスリセット後、ソースノードはデスティネーションノードからResend requestが設定されたレスポンスパケットが送信されるのを待機する。受信したレスポンスパケットのnode_vendor_id, chip __id_hi, chip __id_lo, connection __idを調べて、バスリセット前に使用していたデータと一致した場合、要求されたsequence_numberのsegment データからデータの送信を再開する。

【0180】バスリセット後、コントローラノードはソースノードとデスティネーションノードIDを管理するために、connection_idの管理用テーブルと、connection __idに対応したデータバッファからコネクションが設定されているデスティネーションノードとソースノードのEUI-64の値を読み出して、バスリセットによって変更されたデスティネーションノードとソースノードのノードIDを検出する。

【0181】このノードIDの検出は、例えばバス上の全てのノードに対して、そのノードのROM に書き込まれているEUI-64値をAsynchronousReadトランザクションを使用して読み出すことによって行うことができる。

【0182】各ノードIDの検出が終了すると、コントローラノードはコネクションの再設定のためデスティネーションノードに対してSET DESTINATION コマンドパケットを、ソースノードに対してOBJECT SEND コマンドパケットを送信する。各コマンドパケットを受信したデスティネーションノード及び、ソースノードはそれぞれ対応するレスポンスパケットをコントローラに対して送信する。このときのresponseフィールドはInterim が設定される。

【0183】上記の手順により、バスリセット後にデータ転送及び、コネクションの再設定が行われ、以後のデータ転送及びコネクションの解除は第1の実施の形態と同様の手順で行うことができる。

【0184】本実施の形態では、SET DESTINATION, SET SOURCEコマンドによってデスティネーションノード及びソースノードのEUI-64の値を取得しているが、SET DESTINATION, SET SOURCEコマンドは第1の実施の形態と同様のフォーマットのものを用いて、別のAsynchronousReadトランザクション等を使用してEUI-64の値を取得することも可能である。

【0185】このように、コントローラが同一のバスリセットによって変化しないコネクション識別データをデスティネーションノード及びソースノードに設定することによって、ソースノードとデスティネーションノード間に論理的なコネクションを設定することができ、バスリセットが発生した場合も迅速にデータ転送の再開を行うことができる。

【0186】(第3の実施の形態)次に、図17に示すように、ソースノードと複数のデスティネーションとの間にコネクションを設定してデータ転送を行う場合について説明する。各デスティネーションは異なるサイズの受信バッファを持っている。デスティネーション__1は32KB、デスティネーション__2は48KB、デスティネーション__3は64KBの受信バッファを持つ。また、それぞれのmax __rec 値は、デスティネーション__1が512byte, デスティネーション__2が1024byte, デスティネーション__3が1024byteを示している。

【0187】図18を用いて、本実施の形態のデータ転

31

送の手順について説明する。コントローラはユーザーが選択したデスティネーションノードに対して順次、コネクションを設定するためのSET DESTINATION コマンドパケットを送信する。このコマンドパケットは第1の実施の形態に示したパケットと同様のフォーマットである。各デスティネーションノードに対して送信されるSET DESTINATION コマンドパケットのnode_vendor_id, chip_id_hi, chip_id_lo, connection_idフィールドは全て同じ値がセットされている。

【0188】SET DESTINATION コマンドパケットを受信した各デスティネーションノードはパケットからコネクション識別データを自ノード内のバッファにセットして、コントローラノードに対して第1の実施の形態と同様にSET DESTINATION レスポンスパケットを送信する。

【0189】このとき、デスティネーション_1は図7中のmax_rec フィールドに512byteを示すデータを、buffer_sizeフィールドに32KBを示すデータをセットして、Interimのレスポンスパケットを送信する。

【0190】同様に、デスティネーション_2はmax_rec フィールドに1024byteを示すデータを、buffer_sizeフィールドに48KBを示すデータを、デスティネーション_3はmax_rec フィールドに1024byteを示すデータを、buffer_sizeフィールドに64KBを示すデータをセットして、Interimのレスポンスパケットを送信する。

【0191】各デスティネーションノードからInterimのSET DESTINATION レスポンスパケットを受信した後、コントローラノードはユーザーが選択したソースノードに対して、コネクションを設定するためのSET SOURCEコマンドパケットを送信する。

【0192】このSET SOURCEコマンドパケットは、第1の実施の形態に示したパケットと同様のフォーマットである。このパケットには上記各デスティネーションノードに設定したnode_vendor_id, chip_id_hi, chip_id_lo, connection_idが書き込まれている。max_rec フィールドには、各デスティネーションから受信したmax_rec 値の中で最小のデータをセットする。

【0193】本実施の形態の場合、3つのデスティネーションから受信したmax_rec が示す値はそれぞれ512byte, 1024byte, 1024byte であるので、512byteを示すデータをセットする。また、buffer_sizeフィールドにも各デスティネーションから受信したbuffer_size値の中で最小のデータをセットする。本実施の形態の場合、3つのデスティネーションから受信したbuffer_sizeが示す値はそれぞれ32KB, 48KB, 64KBであるので、32KBを示すデータをセットする。number_of_destinationsフィールドにはデスティネーションの数を示す3がセットされる。

【0194】上記SET SOURCEコマンドパケットを受け取ったソースノードは、コントローラノードにSET SOURCE レスポンスパケットを送信する。SET SOURCEレスポンス

32

パケットは第1の実施の形態に示したパケットと同様のフォーマットであり、第1の実施の形態と同様に各フィールドにデータを設定してレスポンスパケットを送信する。

【0195】ソースノードからAcceptedのSET SOURCEレスポンスパケットを受信した後、コントローラノードはSET_SOURCEレスポンスパケットのstatus_infoフィールドを調べてSuccessであることを確認すると、ソースノードに対してOBJECT SENDコマンドパケットを送信する。OBJECT SEND コマンドの動作については第1の実施の形態と同様であるので説明を省略する。

【0196】以上のAsynchronousトランザクションをコントローラノード、各デスティネーションノード、ソースノードの間で行うことにより、ソースノードと複数のデスティネーションノード間のコネクションの設定が完了し、オブジェクトデータのトランザクションが開始される。上記SET DESTINATION, SET SOURCE, OBJECT SENDパケットはそれぞれ図3(a)のdestination_IDフィールドに受信ノードのノードIDを指定したパケットを使用して各ノードから送信が行われる。

【0197】OBJECT SEND コマンドに従って、第1の実施の形態と同様にオブジェクトデータの転送が開始される。第1の実施の形態と同様にソースノードはdestination_idフィールドにマルチキャスト用のIDを設定したコネクション識別データを含むsegment データパケットをAsynchronous送信する。各デスティネーションノードは受信したマルチキャスト用IDを持つパケットのコネクション識別データを判別し、自ノードに設定されているコネクション識別データと一致する場合segment データを内部の受信バッファに順次書き込んでいく。

【0198】ソースノードは第1の実施の形態と同様に、送信したsegment データのトータルサイズをカウントし、コントローラからSET SOURCEコマンドで通知されたデスティネーションのバッファサイズに達するまで順次データの送信を行う。

【0199】この場合の各パケットのcontrol_flagsフィールドにはNormal data がセットされる。送信データのトータルサイズがデスティネーションのバッファサイズを超えない適当なサイズに達した場合、そのパケットのcontrol_flags フィールドにBuffer endをセットしてデータを送信する。

【0200】図17の例では、トータルサイズがSET SOURCEコマンドパケットで通知されたデスティネーションのバッファサイズと等しくなるsequence number が127のsegmentの送信時に、control_flags をBuffer endにセットする。その後、各デスティネーションノードからレシーブレスポンスパケットが送信されるのを待機する。

【0201】各デスティネーションノードはデータパケットを受信するとパケット中のcontrol_flags を調べ

33

て、この値がNormal data の場合は次のデータパケットが送信されるのを待つ。control __flags の値がBuffer endの場合、受信用バッファに書き込んだデータを別のバッファ（ハードディスク等）にコピーして、バッファをクリアした後に、図12に示すフォーマットのレシーブレスポンスパケットをマルチキャスト用IDを使用して送信する。このとき、control __flags にはReceive success を、sequence number にはBuffer endが設定されたデータパケットのsequence number をセットする。図17の例では127 がセットされる。

【0202】control __flags にBuffer endをセットしたsegment データを送信した後、ソースノードは各デスティネーションノードからレシーブレスポンスパケットを受信して第1の実施の形態と同様に、続きのsegment データの転送を再開する。オブジェクトの最終segment データの送信時には、第1の実施の形態と同様にcontrol __flags フィールドにObject endをセットして同様にデスティネーションからのレシーブレスポンスを待機する。

【0203】各デスティネーションはObject endがセットされたsegment データを受信すると、同様にレシーブレスポンスを送信する。Object endに対するレシーブレスポンスをマルチキャスト用IDを使用して送信した後、各デスティネーションノードはコントローラノードに対して第1の実施の形態と同様にSET DESTINATION レスポンスパケットを送信する。

【0204】また、各デスティネーションからObject endのsegment データに対するレシーブレスポンスパケットを受信した後、ソースノードは第1の実施の形態と同様にコントローラノードに対してOBJECT SEND レスポンスパケットを送信する。上述の手順でソースノードと複数のデスティネーションノードの間のオブジェクト転送が完了する。

【0205】ソースノードと各デスティネーションノードからそれぞれOBJECT SEND, SET DESTINATION のレスポンスパケットを受信した後、第1の実施の形態と同様にコントローラノードは各デスティネーションノードとソースノードに対してコネクションを解除するためのCLEAR CONNECTIONコマンドパケットを送信する。このCLEAR CONNECTIONコマンドパケットのは第1の実施の形態で示したパケットと同様のフォーマットである。

【0206】CLEAR CONNECTIONコマンドパケットを受信した各デスティネーションノードとソースノードは第1の実施の形態と同様の手順でコネクションの解除を行って、CLEAR CONNECTIONレスポンスパケットをコントローラノードにそれぞれ送信する。

【0207】各デスティネーションノード、ソースノードからAcceptedのCLEAR CONNECTIONレスポンスパケットを受信したコントローラノードは、コネクションを解除したconnection_idフラグレジスタの使用中のビットを

34

クリアする。以上の手順で、ソースノードと複数のデスティネーションノードの間に設定された論理的なコネクションが解除され、全てのトランザクションが終了する。

【0208】なお、第3の実施の形態では、第1の実施の形態と同様に、図3(b) に示すパケットフォーマットを用いてオブジェクトデータのトランザクションを行うこともできる。

【0209】また、第3の実施の形態において、オブジェクトデータの転送中にバスリセットが発生しても、各デスティネーションノードが第2の実施の形態と同様の処理を行うことによって、オブジェクトデータの転送を再開させることもできる。この場合、ソースノードは各デスティネーションノードのレシーブレスポンスパケットを受信し、最小となるシーケンス番号を加えた番号のセグメントデータから転送を再開する。また、コントローラは、ソースノードと全てのデスティネーションノードのノードIDを問い合わせ管理する。

【0210】このように、コントローラが同一のコネクション識別データを複数のデスティネーションノード及びソースノードに設定することによって、ソースノードと複数のデスティネーションノード間に論理的なコネクションを設定することができ、データ転送処理はコントローラが介在しない、ソース・デスティネーション間のみのトランザクションで行うことができる。

【0211】また、各デスティネーションの持つ受信能力が異なる場合でも、その中の最も低い能力にあわせてデータ転送を行うことにより、複数のデスティネーションに対してデータ転送を行う場合も、煩雑な処理を必要としないデータ転送を行うことができる。

【0212】（本実施の形態の他の実施形態）本実施の形態は複数の機器（例えば、ホストコンピュータ、インタフェース機器、リーダ、プリンタ等）から構成されるシステムに適用しても1つの機器からなる装置に適用しても良い。

【0213】また、上述した実施形態の機能を実現するように各種のデバイスを動作させるように、上記各種デバイスと接続された装置あるいはシステム内のコンピュータに対し、上記実施形態の機能を実現するためのソフトウェアのプログラムコードを供給し、そのシステムあるいは装置のコンピュータ（CPUあるいはMPU）に格納されたプログラムに従って上記各種デバイスを動作させることによって実施したものも、本実施の形態の範疇に含まれる。

【0214】また、この場合、上記ソフトウェアのプログラムコード自体が上述した実施形態の機能を実現することになり、そのプログラムコード自体、およびそのプログラムコードをコンピュータに供給するための手段、例えばかかるプログラムコードを格納した記憶媒体は本実施の形態を構成する。かかるプログラムコードを記憶

する記憶媒体としては、例えばフロッピーディスク、ハードディスク、光ディスク、光磁気ディスク、CD-ROM、磁気テープ、不揮発性のメモ리카ード、ROM等を用いることができる。

【0215】また、コンピュータが供給されたプログラムコードを実行することにより、上述の実施形態の機能が実現されるだけでなく、そのプログラムコードがコンピュータにおいて稼働しているOS（オペレーティングシステム）あるいは他のアプリケーションソフト等の共同して上述の実施形態の機能が実現される場合にもかかるプログラムコードは本実施の形態の実施形態に含まれることは言うまでもない。

【0216】さらに、供給されたプログラムコードがコンピュータの機能拡張ボードやコンピュータに接続された機能拡張ユニットに備わるメモリに格納された後、そのプログラムコードの指示に基づいてその機能拡張ボードや機能拡張ユニットに備わるCPU等が実際の処理の一部または全部を行い、その処理によって上述した実施形態の機能が実現される場合にも本実施の形態に含まれることは言うまでもない。

【0217】

【発明の効果】上記説明したように、本発明においては、従来の通信方式による不便利性を解決することができる効果が得られる。また、リアルタイム性を必要としないデータの転送においても、高速にデータを転送することを簡便に可能とする効果が得られる。また、本発明の他の特徴によれば、通信帯域をあまり使用しない場合に、多数の通信を同時に行なうことができる効果が得られる。また、本発明の他の特徴によれば、データ転送中断により失われたデータを容易に検出することが可能であるとともに、該データ転送の中断からの復帰を、確実に、かつ、簡単に行なうことができる効果が得られる。また、本発明によれば、複数のコントロール間でコネクション識別データが重複しないように調整する必要がないので、コントローラは、簡単に確実にコネクションを設定できる効果が得られる。また、本発明の他の特徴によれば、複数のコントロールノードが個別に複数の論理的コネクションをソース、デスティネーション間に設定した場合も、個々のノードは、コネクションを設定したコントローラを上記ノード固有の情報であるEUI-64などの固有のノードIDにて判別することが可能となるので、個々のノードは、確実に論理的コネクションを識別できる効果が得られる。本発明の他の特徴によれば、1個のコネクション識別データにより、容易に複数のデスティネーションに対してデータを単一セグメントパケットで送信することができるため、バス上のトラフィックを低減する効果が得られる。また、本発明の他の特徴によれば、それぞれのデスティネーションの受信バッファが異なっている、ソースは、それぞれのコントローラから指示された単一のバッファサイズのみを管理し

てデータ送信を行うことができるため、実装が容易となる効果が得られる。

【図面の簡単な説明】

【図1】第1の実施の形態の実施の形態を表すブロック図である。

【図2】第1の実施の形態に係るデータ転送動作の概要を説明するための図である。

【図3】Asynchronous トランザクションに使用されるパケットを示す図である。

10 【図4】第1の実施の形態に係る各ノード間のコマンドやデータの授受のダイアグラムを示す図である。

【図5】第1の実施の形態のコマンドパケットに用いられるデータフォーマットを示す図である。

【図6】第1の実施の形態のコマンドパケットに対するレスポンスパケットに用いられるデータフォーマットを示す図である。

【図7】第1の実施の形態においてデスティネーションに対して設定を行うコマンドパケットに用いられるデータを示す図である。

20 【図8】第1の実施の形態においてソースに対して設定を行うコマンドパケットに用いられるデータを示す図である。

【図9】第1の実施の形態のソースに対してデータ送信の指示を行うコマンドパケットに用いられるデータを示す図である。

【図10】第1の実施の形態のオブジェクトデータの転送を説明するための図である。

【図11】第1の実施の形態のオブジェクトデータの送信パケットのフォーマットを示す図である。

30 【図12】第1の実施の形態のオブジェクトデータの受信確認パケットのフォーマットを示す図である。

【図13】第1の実施の形態のコネクション解除を行うコマンドパケットに用いられるデータを示す図である。

【図14】第2の実施の形態に係る各ノード間のコマンドやデータの授受を説明するダイアグラムを示す図である。

【図15】第2の実施の形態においてデスティネーションに対して設定を行うコマンドパケットに用いられるデータを示す図である。

40 【図16】第2の実施の形態においてソースに対して設定を行うコマンドパケットに用いられるデータを示す図である。

【図17】第3の実施の形態のオブジェクトデータの転送を説明するための図である。

【図18】本発明の第3の実施の形態に係る各ノード間のコマンドやデータの授受を説明するダイアグラムを示す図である。

【図19】第1の実施の形態の情報を管理用テーブルの説明図である。

50 【図20】第2の実施の形態の情報を管理用テーブルの

説明図である。

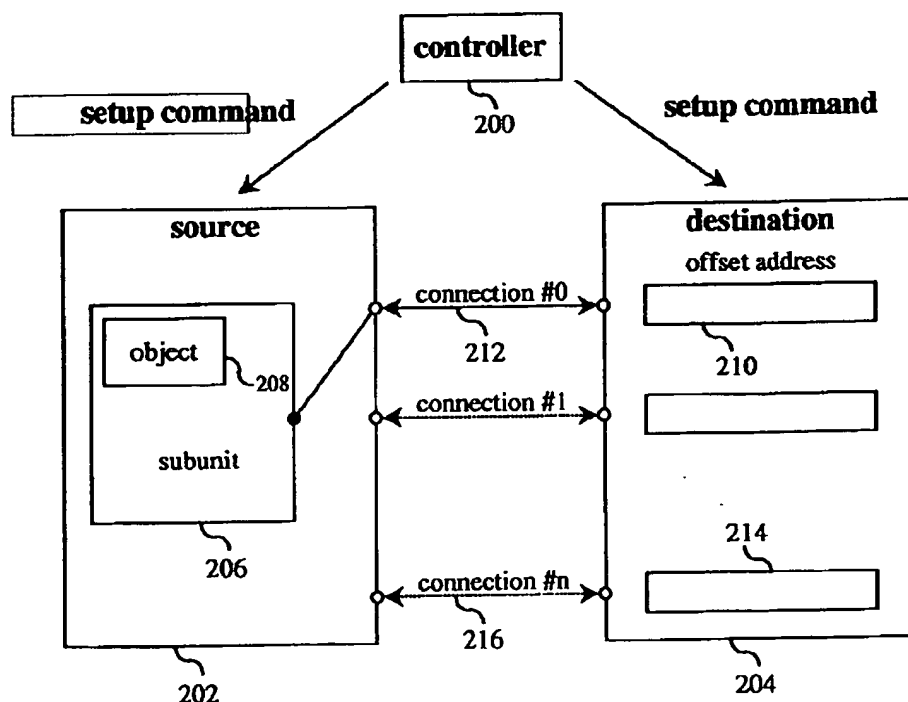
【図21】従来例の説明図である。

【符号の説明】

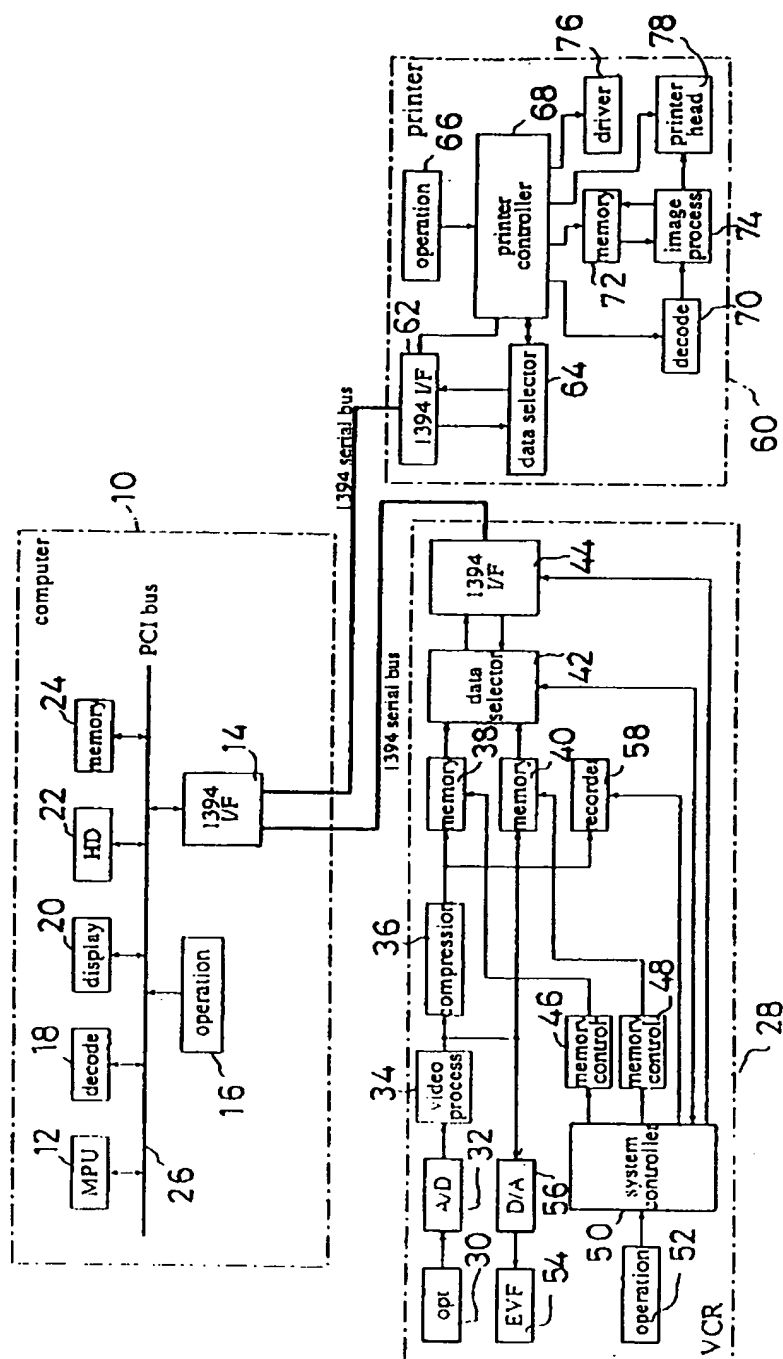
- 10 computer
- 12 演算処理装置(MPU)
- 14 第一の1394インターフェイス
- 16 キーボードなど第一の操作部
- 18 第一のデコーダ
- 20 CRT ディスプレイなどの表示装置
- 22 ハードディスク
- 24 第一のメモリ
- 26 PCI バスなどのコンピュータ内部バス
- 28 VCR
- 30 撮像光学系
- 32 A/D 変換器
- 34 ビデオ処理部
- 36 圧縮伸長回路
- 38 第一のメモリ
- 40 第二のメモリ
- 42 第一のデータセクタ
- 44 第二の1394インターフェイス
- 46 第一のメモリ制御回路
- 48 第二のメモリ制御回路
- 50 システムコントローラ

- 52 第二の操作部
- 54 電子ビューファインダ
- 56 D/A 変換器
- 58 記録部
- 60 プリンタ
- 62 第三の1394インターフェイス
- 64 第二のデータセクタ
- 66 第三の操作部
- 68 プリンタコントローラ
- 10 70 第二のデコーダ
- 72 第三のメモリ
- 74 画像処理部
- 76 ドライバ
- 78 プリンタヘッド
- 200 コントロールノード
- 202 ソースノード
- 204 デスティネーションノード
- 206 ソースノード内部のサブユニット
- 208 画像データ等のobject
- 20 210 デスティネーションノード内部の第一のメモリ空間
- 212 第一のコネクション
- 214 デスティネーションノード内部の第n のメモリ空間
- 216 第nのコネクション

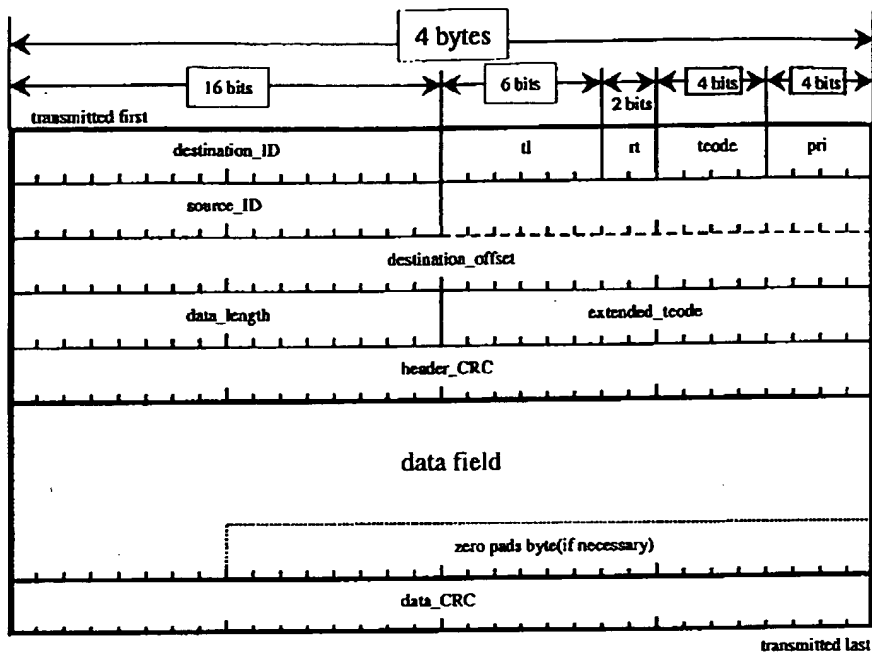
【図2】



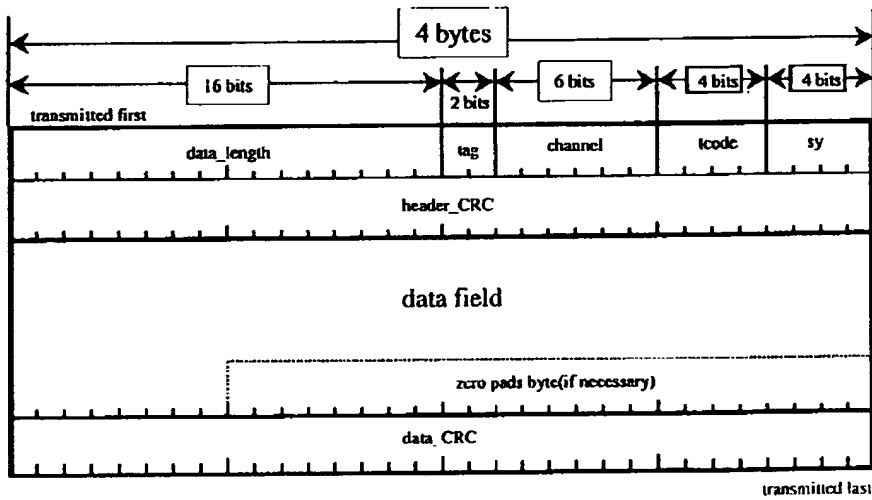
【図1】



【図3】

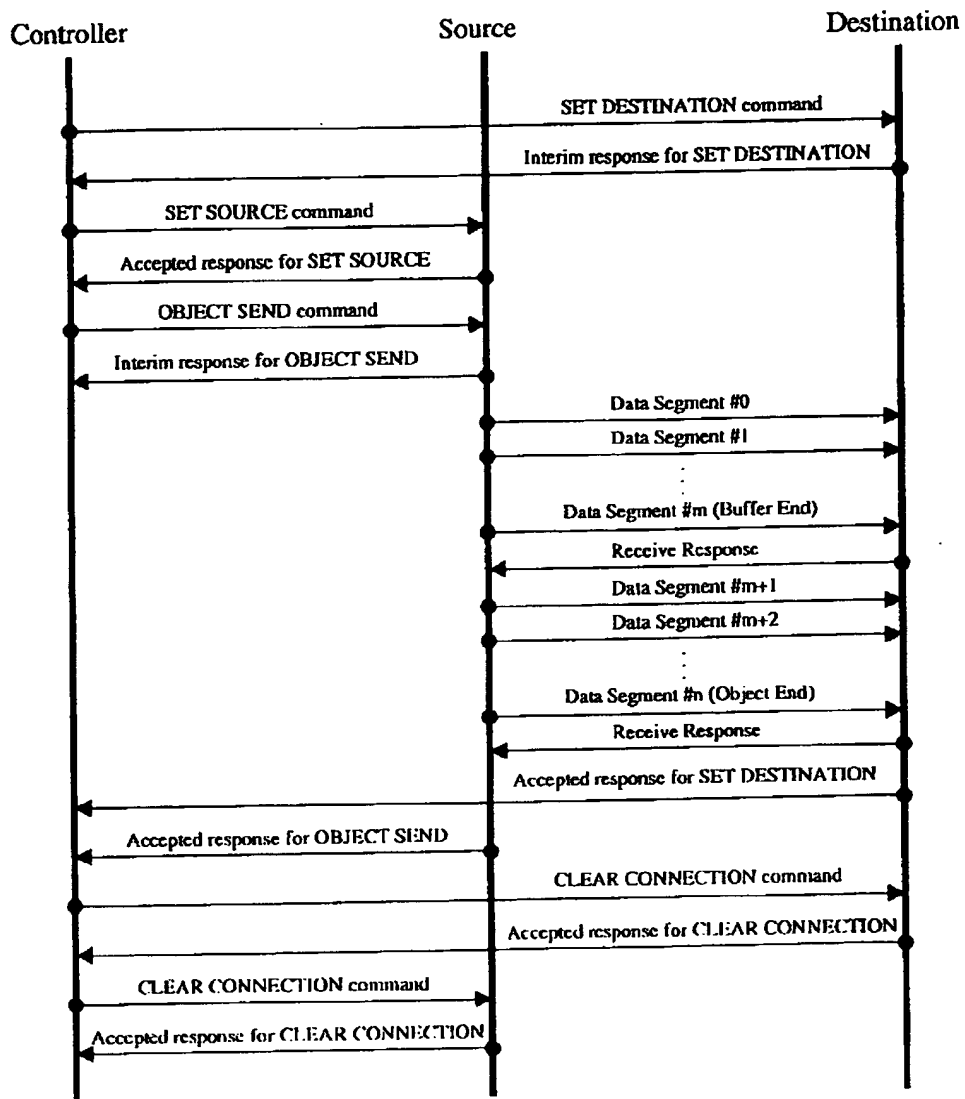


(a)

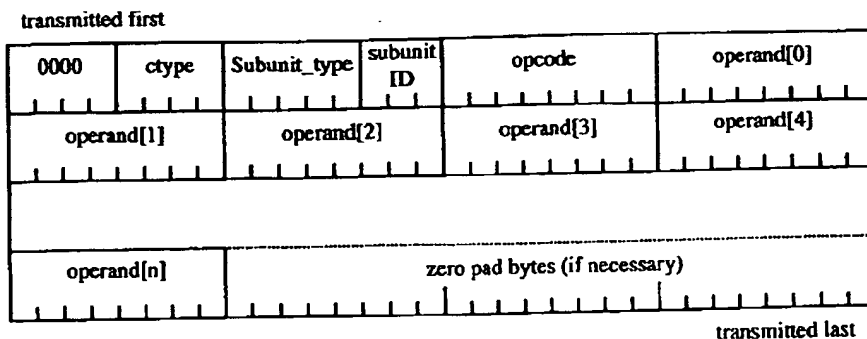


(b)

【図4】



【図5】



transmitted first

0000	response	Subunit_type	subunit ID	opcode	operand[0]
operand[1]		operand[2]		operand[3]	operand[4]
operand[n]		zero pad bytes (if necessary)			
transmitted last					

【図7】

	msb						lsb
opcode	SET_DESTINATION						
operand[0]	node_vendor_id						
operand[1]							
operand[2]							
operand[3]	chip_id_hi						
operand[4]	chip_id_lo						
operand[5]							
operand[6]							
operand[7]	connection_id						
operand[8]							
operand[9]	reserved			max_rec			
operand[10]	buffer_size						
operand[11]							
operand[12]	status_info						
operand[13]							

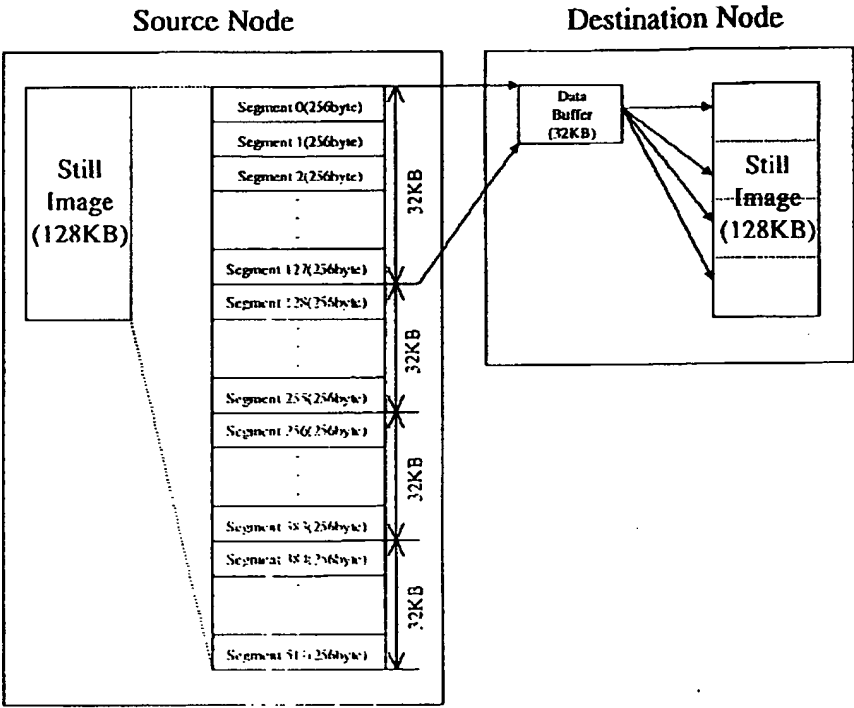
【図9】

	msb							lsb
opcode	OBJECT SEND							
operand[0]								
operand[1]	node_vendor_id							
operand[2]								
operand[3]	chip_id_hi							
operand[4]								
operand[5]	chip_id_lo							
operand[6]								
operand[7]								
operand[8]	connection_id							
operand[9]	subfunction							
operand[10]	status_info							

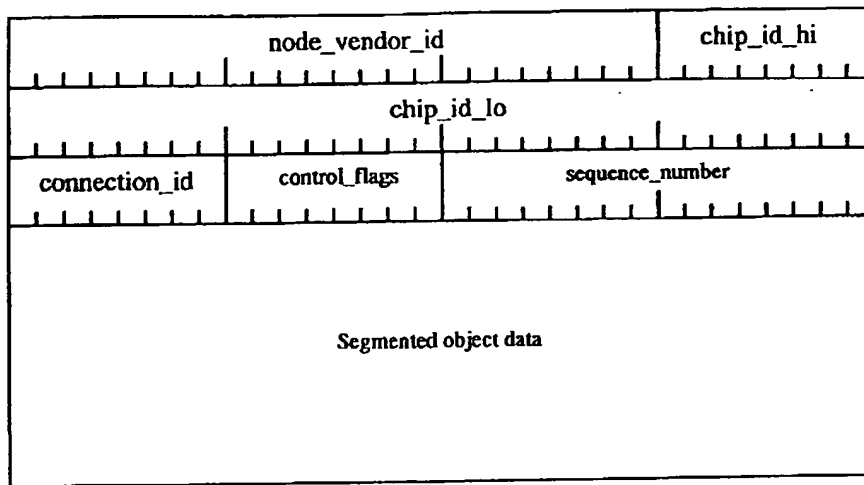
【図8】

	msb							lsb
opcode	SET_SOURCE							
operand[0]	node_vendor_id							
operand[1]								
operand[2]								
operand[3]	chip_id_hi							
operand[4]	chip_id_lo							
operand[5]								
operand[6]								
operand[7]								
operand[8]	connection_id							
operand[9]	reserved				max_rec			
operand[10]	buffer_size							
operand[11]								
operand[12]								
operand[13]	number_of_destinations							
operand[14]	status_info							

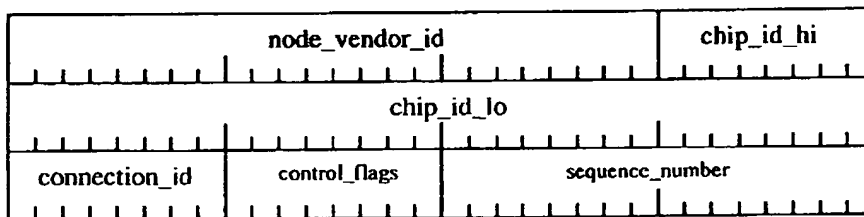
【図10】



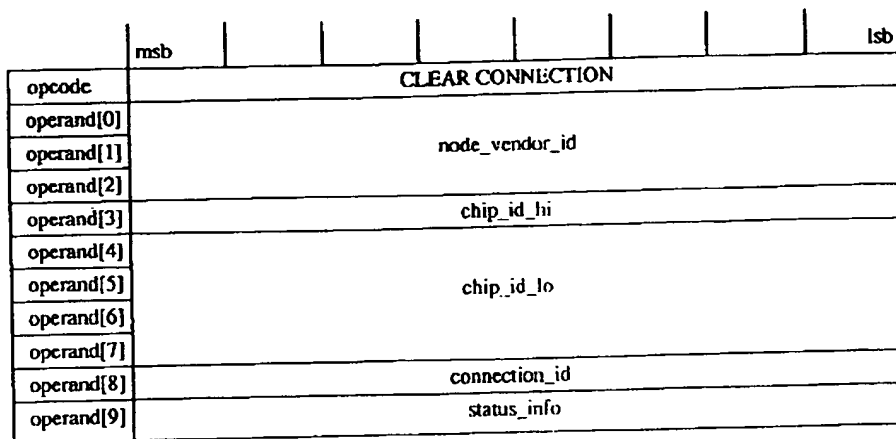
【図11】



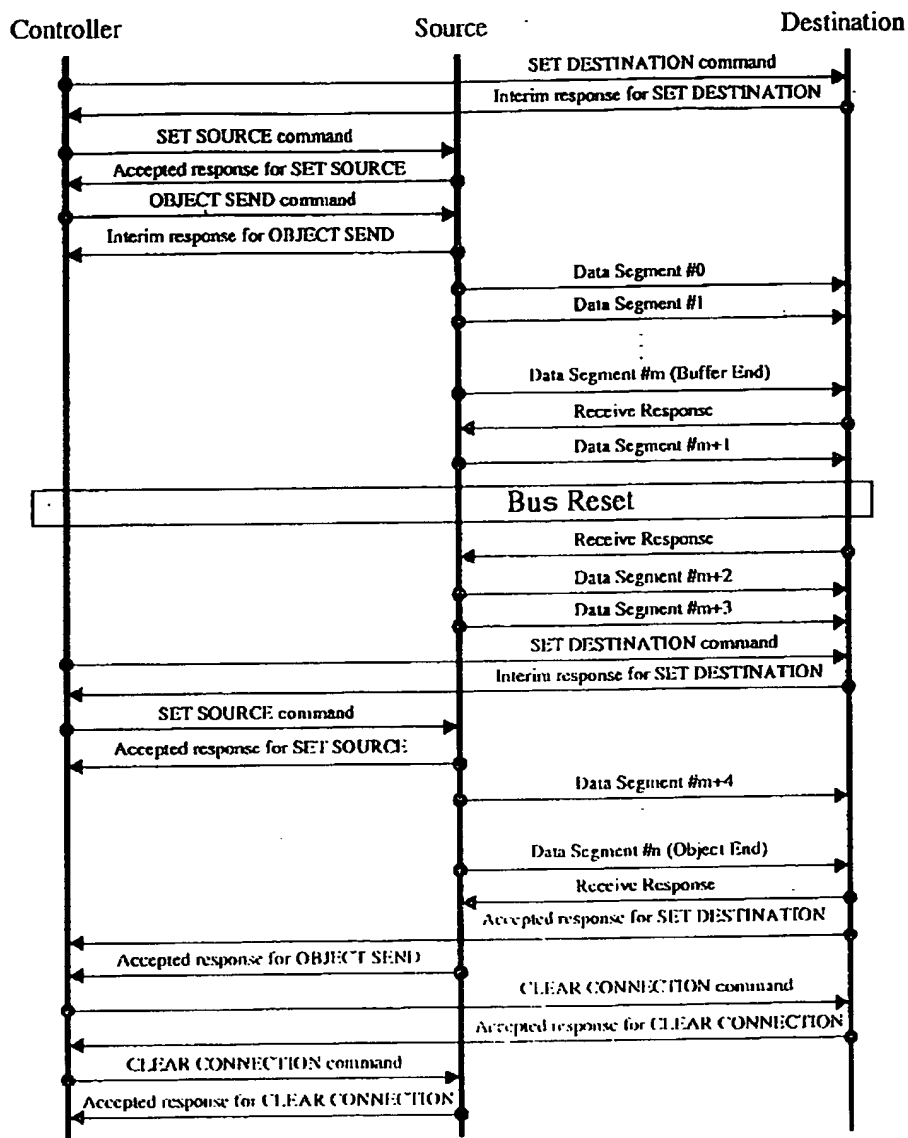
【図12】



【図13】



【図14】



【図19】

connection_id	flag	max_rec	buffer_size
0	1	512	32KB
1	0	0	0
...			
n-1	0	0	0
n	0	0	0

【図15】

	msb						lsb
opcode	SET DESTINATION						
operand[0]	node_vendor_id						
operand[1]							
operand[2]							
operand[3]	chip_id_hi						
operand[4]	chip_id_lo						
operand[5]							
operand[6]							
operand[7]							
operand[8]	connection_id						
operand[9]	destination_node_vendor_id						
operand[10]							
operand[11]							
operand[12]	destination_chip_id_hi						
operand[13]	destination_chip_id_lo						
operand[14]							
operand[15]							
operand[16]							
operand[17]	reserved			max_rec			
operand[18]	buffer_size						
operand[19]							
operand[20]							
operand[21]	status_info						

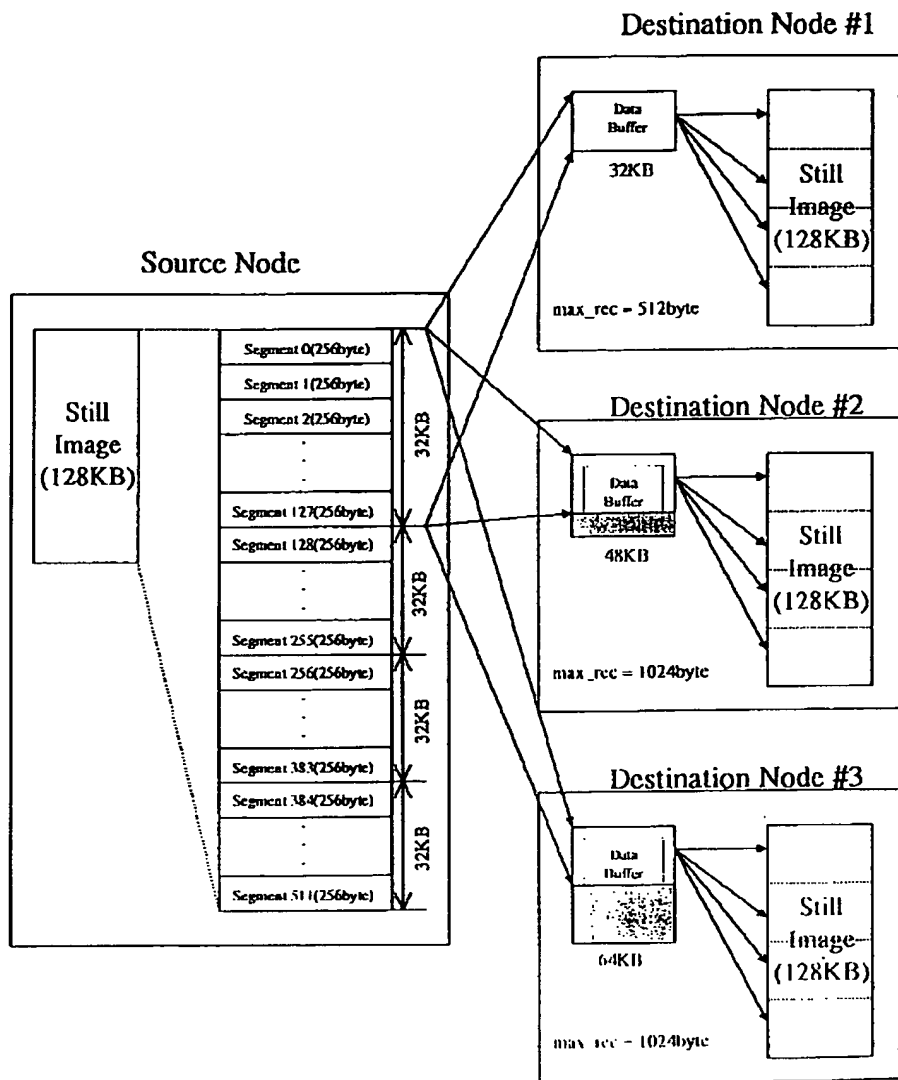
【図20】

connection_id	flag	max_rec	buffer_size	number_of_destinations	EUI-64	
0	1	512	32KB	1	Source	xxxxxx xx xxxxxxxx
1	0	0	0	0	Destination #1	yyyyyy yy yyyyyyyy
n-1	0	0	0	0		
n	0	0	0	0		

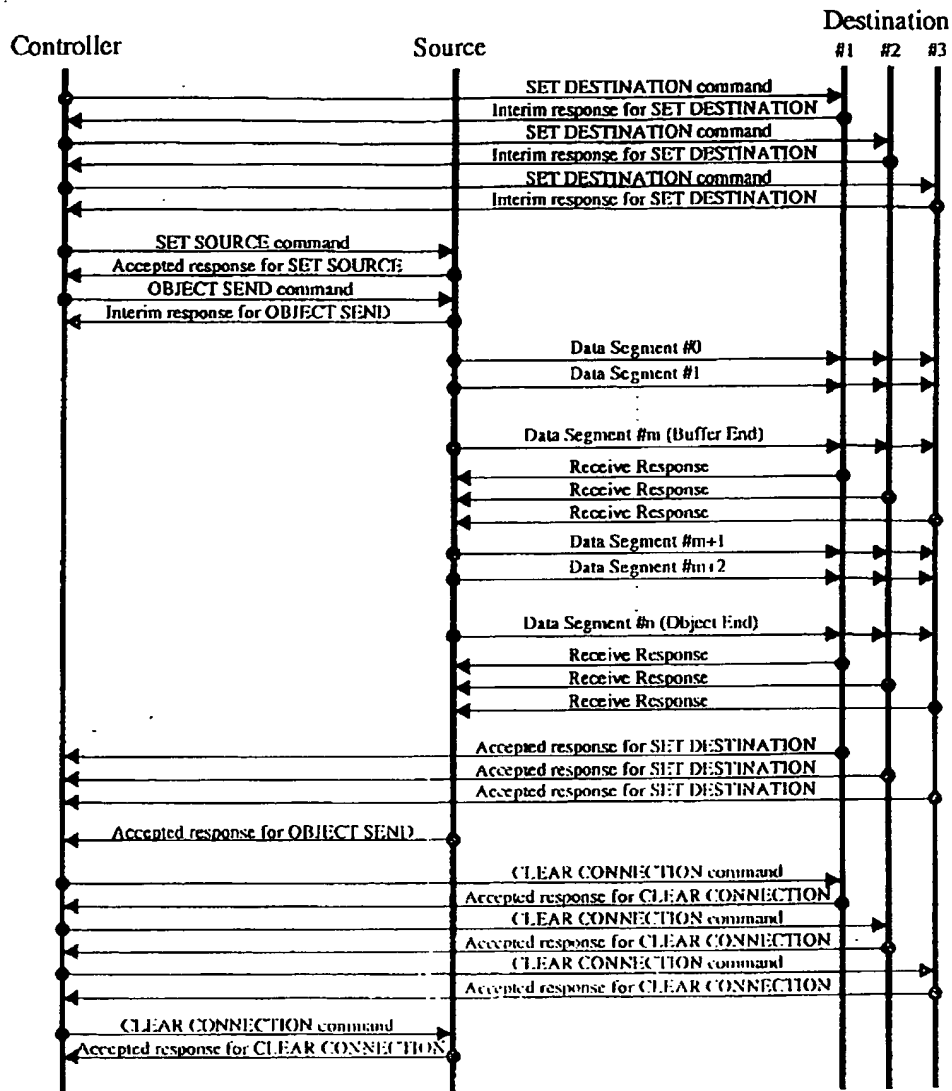
【図16】

[illegible]

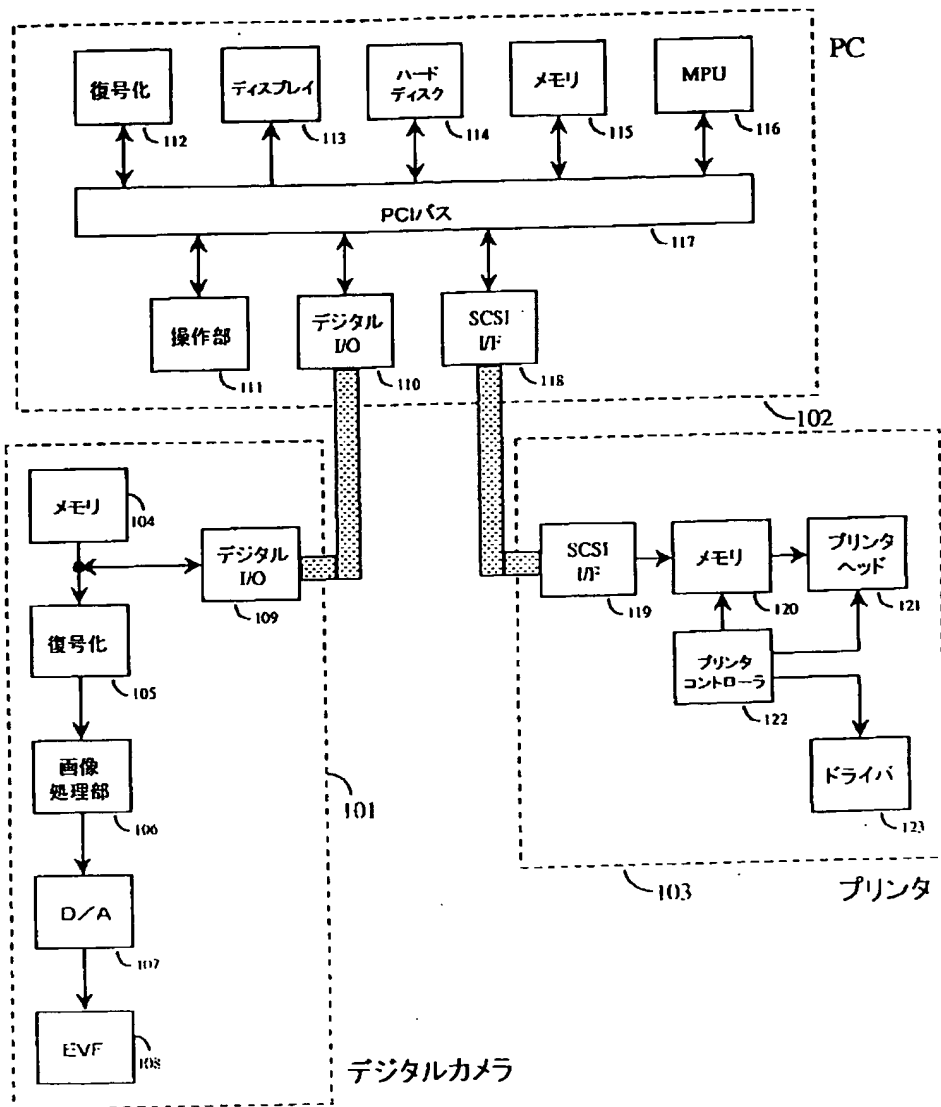
【図17】



【図18】



【図21】



フロントページの続き

(72)発明者 小林 崇史
 東京都大田区下丸子3丁目30番2号 キヤ
 ノン株式会社内